VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ANONYMITA V P2P SÍTÍCH

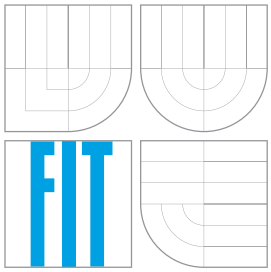
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ADAM BRUNAI

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ANONYMITA V P2P SÍTÍCH

ANONYMITY IN P2P NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ADAM BRUNAI

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MAROŠ BARABAS

BRNO 2014

Abstrakt

Svoboda slova a právo na soukromí patří mezi nejdůležitější součásti moderní společnosti, ale přesto jsou často porušována. Také z tohoto důvodu se tato práce zabývá modelmi P2P sítí, anonymizačními technikami, technikami zabezpečujícími necenzurovatelnost a jejich použitím v prostředí reálných P2P sítí a publikačních systémů. Budeme diskutovat jejich efektivitu a vhodnost pro konkrétní účely, ale také bezpečnostní aspekty jejich použití. V druhé polovině práce bude představen publikační protokol LSPP, který je implementací anonymní a necenzurovatelné P2P sítě použitelný jako knihovna. Na závěr práce analyzujeme navržený protokol a porovnáme ho se stávajícími řešeními.

Abstract

Freedom of speech and the right to privacy are maybe the most important elements of a modern society, yet the rights are often violated. This fact was the main reason for writing this thesis covering P2P network models, anonymity, censorship resistance and their use in real P2P networks and publishing systems. We discuss their effectiveness and suitability for specific purposes, but also the security considerations of their use. The second part of this thesis presents the LSPP publishing protocol, which is an library implementation of an anonymous censorship resistant P2P network. Finally, we analyze the proposed protocol and compare it with existing solutions.

Klíčová slova

P2P, anonymita, necenzurovatelnost, bezpečnost, DoS, šifrování, sdílení dat, Mixes, Onion Routing, Crowds, DC-Net, Chord, LSPP, publikační protokol

Keywords

P2P, anonymity, censorship resistance, security, DoS, encryption, data sharing, Mixes, Onion Routing, Crowds, DC-Net, Chord, LSPP, publishing protocol

Citace

Adam Brunai: Anonymita v P2P sítích, diplomová práce, Brno, FIT VUT v Brně, 2014

Anonymita v P2P sítích

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Maroša Barabasa. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Adam Brunai
26. května 2014

Poděkování

Rád bych poděkoval vedoucímu této práce panu Ing. Marošovi Barabasovi za pomoc při výběru tématu diplomové práce, odborné konzultace, rady a připomínky.

Obsah

Úvod

Vývoj počítačových systémov v posledných desaťročiach prešiel dramatickým vývojom. Spolu so zvýšením výkonu nielen osobných počítačov a prenosných zariadení klesli aj náklady na prenos dát v prostredí internetu. Tento trend sa snažia využiť takzvané P2P (Peer-to-peer) siete, ktoré sa stále častejšie dostávajú z oblasti teoretických modelov aj do bežnej praxe. Prednosťou P2P sietí je ich vysoká škálovateľnosť a často i nezávislosť na centralizovaných systémoch, tak ako ich poznáme z architektúry klient-server.

P2P siete však neprinášajú len pozitívne vlastnosti, ale prinášajú aj nové problémy, s ktorými sa musia ich návrhári vysporiadať. V tejto práci sa budeme zaoberať špeciálnym typom P2P sietí, ktorých úlohou je poskytnúť ich používateľom vysokú bezpečnosť, odolnosť voči zásahom tretích strán do obsahu siete (napríklad cenzúra) a zároveň chrániť identitu jej používateľov. Poskytujú teda prostriedky pre zachovanie slobody názoru, slobody slova a súkromia. Tieto hodnoty sú základným stavebným kameňom demokracie a slobodnej spoločnosti ako takej.

Vytvorenie takýchto systémov nám umožňujú nielen prostriedky matematiky a kryptografie, ale predovšetkým ochota používateľov podieľať sa na takomto systéme a prispievať svojimi výpočetnými a sieťovými prostriedkami k fungovaniu siete. Z tohoto dôvodu musíme zohľadňovať pri návrhu P2P siete nielen jej teoretické aspekty, ale aj výkonnosť navrhnutého systému v reálnej počítačovej sieti. Práve preto sa budeme v tejto práci zaoberať odpoveďami na otázky: Aké existujú spôsoby zachovania anonymity užívateľa? Sú tieto techniky rovnako efektívne? Je možné zabezpečiť necenzúrovateľnosť v prostredí P2P sietí? A v neposlednom rade, aké prostriedky zvoliť k zaisteniu bezpečnosti prenášaných dát?

Súčasťou tejto práce je návrh anonymného a necenzúrovateľného publikačného systému, ktorý umožňuje flexibilnú výmenu textových informácií a zároveň splňuje vysoké nároky na bezpečnosť. Dôležitou vlastnosťou tohoto systému bude predovšetkým zachovanie vysokej efektivity prenosu dát a preto budeme musieť prijať určité kompromisy medzi mierou anonymity a výkonnosťou. Rôzne techniky, ale taktiež spôsoby ich implementácie, budeme rozoberať v nasledujúcich kapitolách tejto práce.

Obsah práce je členený do piatich kapitol. V prvej kapitole sa pozrieme na kľúčové vlastnosti a typy P2P sietí. V druhej kapitole sa podrobne pozrieme na anonymizačné techniky, ktoré nájdu svoje uplatnenie nie len v P2P sieťach, ale aj v iných komunikačných architektúrach. V tretej kapitole sa plynule presunieme na tématiku bezpečnosti. Tieto poznatky zužitkujeme v posledných dvoch kapitolách. V predposlednej kapitole sa podrobne oboznámime s návrhom publikačného protokolu LSPP, ktorého vlastnosti následne podrobíme analýze v poslednej kapitole tejto práce.

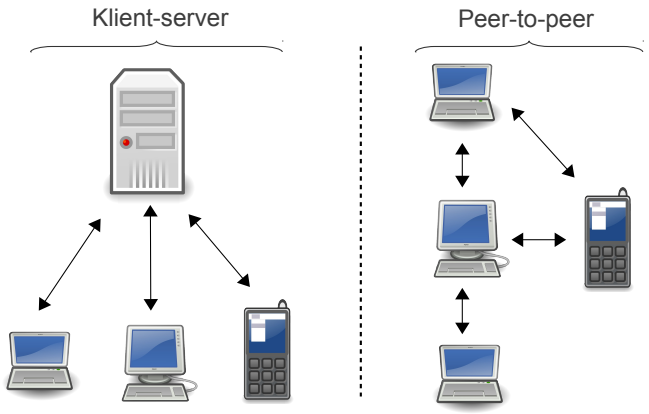
Kapitola 1

Peer-to-peer sieť (P2P)

Už od počiatkov formovania internetu vznikalo viacero konceptov komunikácie medzi jednotlivými užívateľmi siete. Princípy P2P komunikácie sa začali formovať už v sedemdesiatych rokoch 20. storočia s príchodom ARPANETu [19]. Ich popularita však vzrástla až s príchodom aplikácií na zdieľanie súborov (napr. Napster, Gnutella, BitTorrent).

P2P sieť je distribuovaná sieťová architektúra, v ktorej sú užívatelia siete producentmi a zároveň konzumentmi obsahu. Jednotlivé *uzly* siete (peers) sú navzájom rovnocenné a spolupracujú za účelom dosiahnutia spoločného cieľa. Jednotlivé uzly sú navzájom prepojené pomocou *vrstvy prepojení* (overlay network), ktorá je vybudovaná v zásade na existujúcej infraštruktúre. Jedná sa o abstraktnú vrstvu, ktorá je nezávislá od skutočnej topológie siete. Najčastejšie sa môžeme stretnúť s implementáciou P2P sieti v prostredí internetu, avšak model P2P komunikácie je možné použiť na ľubovolné zariadenia schopné vzájomnej komunikácie (napr. senzorové siete, bezdrôtové zariadenia, dopravné prostriedky atď.).

Toto je v priamom kontraste s architektúrou *klient-server* (viď obr. 1.1), v ktorej je postavenie jednotlivých účastníkov komunikácie asymetrické, pričom jednotliví klienti medzi sebou nekomunikujú priamo, ale vždy len prostredníctvom serveru. Pre klient-server architektúru je taktiež typické, že komunikáciu takmer vždy inicializuje klient.



Obr. 1.1: Porovnanie architektúr klient-server a P2P

1.1 Oblasti použitia

P2P siete majú v porovnaní s architektúrou klient-server výhodu v nízkych nákladoch na založenie a prevádzku distribúcie obsahu. K ich nevýhodám patrí naopak nemožnosť zaručiť spoľahlivý a nepretržitý prístup ku všetkým *objektom*[1] sprístupneným v rámci siete. Obecne platí, že obľúbené a často vyžadované objekty majú v sieti lepšiu dostupnosť ako staršie a menej vyžadované.

Na základe týchto vlastností sa ukázalo byť vhodné použitie v nasledujúcich oblastiach [5]:

Zdieľanie súborov – jedná sa o najznámejšie uplatnenie P2P sietí v praktických aplikáciach. Jednotlivé aplikácie na zdieľanie súborov sa líšia vo veľkosti užívateľskej komunity, spoľahlivosti, rýchlosti, ale i architektúre a prístupe ku zdieľaným súborom. Trendom v tomto segmente je výraznejšia decentralizácia a zlepšovanie robustnosti siete. Motiváciou k tomuto kroku je okrem iného osud aplikácie Napster[2].

Hlasové služby (VoP2P) – najvýraznejším reprezentantom tejto kategórie je aplikácia Skype, ktorá sa stala synonymom pre hlasové služby v prostredí internetu. Skype využíva klientský software jednotlivých používateľov na vytvorenie tzv. *superuzlov*, cez ktoré smerujú hovory užívateľov, ktorí nedisponujú verejnou IP adresou [4].

Instant messaging (IM) – do tejto kategórie spadajú komunikačné programy pre rozhovor dvoch alebo viacerých osôb, avšak vzhľadom na nízku náročnosť textovej komunikácie na prenosové pásmo je použitie P2P technológií pre IM vzácne.

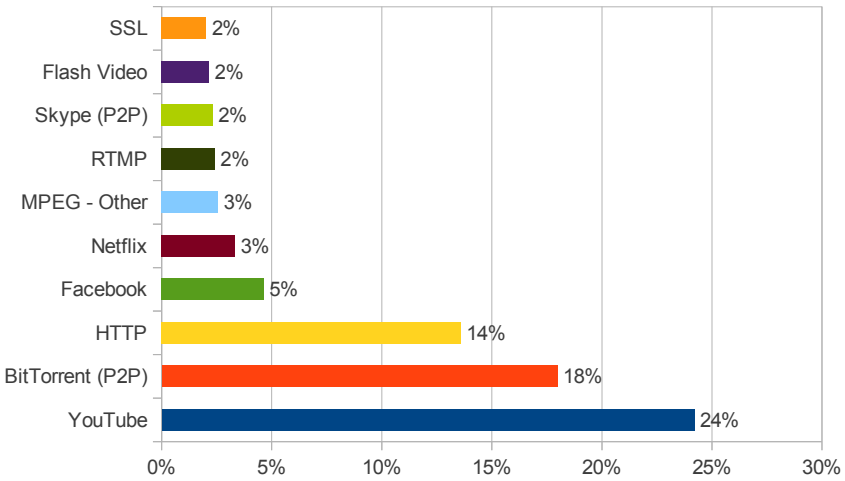
Kryptomeny – vytvorenie decentralizovanej virtuálnej meny je myšlienka stará viac ako 20 rokov, ktorej autorom je Timothy C. May [12]. Skutočné rozšírenie virtuálnej meny založenej na P2P sietach však prišlo až s príchodom populárnej meny Bitcoin. Hlavnou výhodou je nezávislosť celého systému na akejkoľvek centrálnej autorite a nemožnosť regulácie danej meny treťou stranou. V poslednej dobe zaznamenávame príchod ďalších kryptomien, niekedy označovaných aj ako kryptomeny 2. generácie. Jedná sa o meny ako napríklad Litecoin, Peercoin, Namecoin atď.

Multimediálny streaming (P2PTV) – na rozdiel od bezdrôtového televízneho alebo rozhlasového vysielania je prenos multimediálneho streamu v prostredí internetu úloha náročná na konektivitu serveru a jeho systémové prostriedky. Tradičný spôsob spočíva vo vytvorení individuálneho streamu pre každého užívateľa, čo je pre vyšší počet užívateľov problematické. Tento problém je možné čiastočne vyriešiť multicastom na úrovni IP infraštruktúry, avšak len v lokálnych sieťach. Vysielanie multimediaálneho streamu v prostredí P2P siete môže teda nezanedbateľne znížiť nároky na prenos.

Podľa správy [17] publikovanej spoločnosťou Sandvine v roku 2013, ktorá skúmala podiel aplikácií na prenášaných dátach v Európe, tvorili P2P aplikácie až 20 % prenášaných dát. To je mierny pokles oproti predchádzajúcim rokom. Grafické znázornenie môžeme vidieť na Obrázku 1.2.

[1] Objektom sa v kontexte P2P sietí spravidla myslí súbor, ale môže sa jednať o ľubovoľné dáta zdielané v sieti (video stream, reputácia súboru/užívateľa, atď).

[2] Centralizovaná sieť zameraná na distrubúciu mp3 súborov. Bola zrušená v roku 2001 na základe rozhodnutia amerického súdu z dôvodu porušovania autorských práv užívateľmi [1].



Obr. 1.2: Typ dát prenášaných Internetom v Európe

1.2 Architektúra P2P sietí

Pri návrhu P2P sietí sa musia vývojári vysporiadať s množstvom problémov. Hneď prvý je *problém inicializácie siete* (boot-strap problem). Na počiatku formovania siete existuje iba niekoľko málo uzlov, ktoré ešte nemajú vytvorené vzájomné prepojenia a nemajú teda možnosť objaviť susedné uzly. Existuje viacero riešení tohoto problému, pričom neexistuje riešenie, ktoré by zachovávalo úplnú decentralizáciu siete a zároveň bolo dostatočne optimálne pre všetky oblasti použitia P2P sietí. Tradičné riešenie spočíva v definovaní serverov známych pre všetky uzly, ktoré zabezpečia pridanie nových užívateľov do siete.

Pri návrhu P2P siete je nutné splniť nasledujúce požiadavky [5]:

- Vytvorená sieť musí byť *samo-organizovateľná*. Keďže neexistuje žiadna centrálna autorita, uzly siete musia byť schopné spolupracovať za účelom dodržania optimálnej konfigurácie siete a zabezpečiť tak jej dlhodobé fungovanie. Táto úloha musí byť splnená len na základe lokálnych (neúplných) informácií o stave siete.
- Pretože sa počet užívateľov v sieti môže dynamicky meniť, je dôležité zabezpečiť *škálovatelnosť* siete. Existujú operácie, ktoré sú optimálne iba v sieťach určitej veľkosti a ich nesprávne použitie môže degradovať priepustnosť siete. Medzi takéto operácie patrí napríklad vyhľadávanie.
- Každý uzol siete sa chová *autonómne*. To znamená, že každý uzol siete je schopný zistiť svoje vlastné schopnosti na základe ním držaných prostriedkov, vie kedy sa má pripojiť alebo odpojiť zo siete, vie aké požiadavky môže odoslať a ako reagovať na požiadavky ostatných uzlov. Táto vlastnosť so sebou prináša rôzne bezpečnostné riziká.
- Uzly siete *zdieľajú spoločné zdroje*. Môže sa jednať o miesto na disku, výpočtový výkon, internetové pripojenie atď. Ďalej je dôležité, aby boli tie prostriedky zdieľané férovo, čo je možné čiastočne zabezpečiť prednastavením vhodných limitov.

- V neposlednom rade je nutné, aby sieť dokázala reagovať na časté pripojenia a odpojenia užívateľov zo siete. Doba zotrvania užívateľa v sieti sa nazýva v anglickej literatúre *churn* a zavisí od nej dostupnosť objektov v sieti.

1.2.1 Model siete

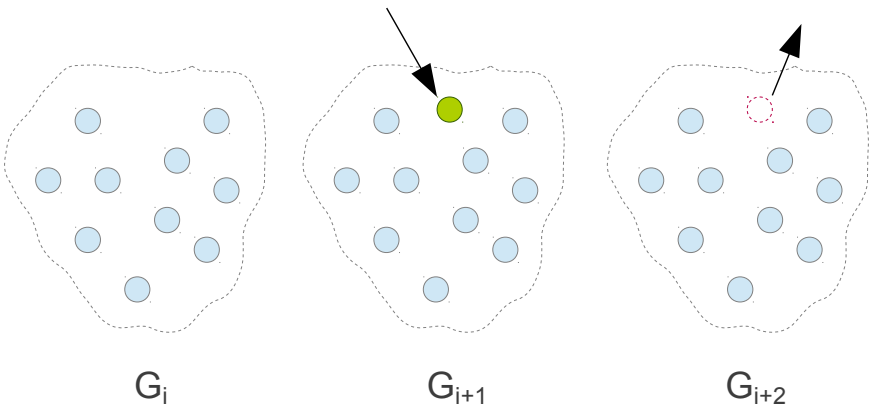
Za účelom upresnenia fungovania P2P sietí v nasledujúcich kapitolách definujeme P2P sieť formálne ako orientovaný graf

$$G = (V, E)$$

kde V je množina uzlov a E je množina prepojení medzi jednotlivými uzlami (hrany) [5].

Každý uzol $p \in V$ má unikátny číselný identifikátor *pid* a sieťovú adresu (napr. IP adresu) *nid*. Pričom $(p, q) \in E$ znamená, že medzi uzlami p a q existuje priame spojenie a môžu si navzájom posielať správy (p môže poslať správu q pomocou *nid* prislúchajúcej q). Vo väčšine modelov P2P sietí je snaha, aby bol graf G spojitý.

Pretože užívatelia P2P siete sa môžu do siete pripájať a odpájať, aproximujeme túto vlastnosť pomocou sekvencie dočasných stavov siete $G_i = (V_i, E_i)$, $G_{i+1} = (V_{i+1}, E_{i+1})$, $G_{i+2} = (V_{i+2}, E_{i+2})$, ... (viď obr. 1.3). Pre uzly $m, n \in V$ pripojenie uzla p' k sieti G_i v čase i spôsobí zmenu stavu siete reprezentovanej grafom $G_{i+1} = (V_{i+1}, E_{i+1})$ kde $V_{i+1} = V_i \cup \{p'\}$ a $E_{i+1} = E_i \cup \{(p', m)\} \cup \{(n, p')\}$. To znamená, že uzol p' bol pridaný do množiny uzlov, pričom existuje minimálne jedno odchádzajúce a jedno prichádzajúce spojenie k uzlu p'. Obdobne pre odpojenie uzla zo siete reprezentovanej grafom G_i v čase i platí, že $G_{i+1} = (V_{i+1}, E_{i+1})$ kde $V_{i+1} = V_i - \{p'\}$ a $\forall m, n : E_{i+1} = E_i - \{(p', m)\} - \{(n, p')\}$. Operácie pripojenia do siete a odpojenia zo siete budeme po rade nazývať *join* a *leave*.



Obr. 1.3: Zmeny stavu siete pri pripojení a odpojení uzla

Uzly p a q sú navzájom *susedné*, ak platí $(p, q) \in E \vee (q, p) \in E$. Inak povedané, musí medzi nimi existovať priame spojenie. Operácie pripojenia a odpojenia zo siete priamo ovplyvňujú *smerovacie tabuľky*[3] susedných uzlov a nepriamo môžu taktiež ovplyvňovať ostatné uzly. P2P sieť sa musí s týmito zmenami vysporiadať pomocou výmeny informácií o G medzi jednotlivými uzlami. Tento mechanizmus nazývame *údržba siete* (network maintenance).

[3]Smerovacia tabuľka plní v P2P sieťach podobnú úlohu ako smerovacia tabuľka na vrstve L3, avšak funguje na aplikačnej úrovni.

V niektorých modeloch je na množine E definovaná relácia usporiadania a teda pre každé dva uzly p a q platí, že sú susedné vtedy, ak pid_p a pid_q sú vo vzťahu predchodca alebo nasledovník.

1.2.2 Smerovanie

Je dôležité si uvedomiť, že model siete uvedený v predchádzajúcej sekcii je iba aproximácia. Môže totiž nastať situácia, kedy sa do siete pripojí viacero uzlov alebo sa naopak viacero uzlov odpojí. To môže vyústiť až v rozdelenie siete na viacero oddelených segmentov. Preto je veľmi dôležité riešiť tento problém už v procese návrhu siete. Jedným z kritických mechanizmov je *smerovanie*.

Uvedený graf G je globálny pohľad na P2P sieť, jednotlivé uzly však disponujú len lokálnymi informáciami získanými od susedných uzlov. Na základe týchto informácií si vytvárajú smerovaciu tabuľku $R_p = (V_p, E_p) \subseteq G$, ktorú definujeme nasledovne

$$\begin{aligned} V_p &\subseteq V, \\ E_p &\subseteq E, \\ E_p &\equiv \forall (p,q) \in E \Rightarrow (p,q) \in E_p, \\ V_p &\equiv \forall (p,q) \in E \Rightarrow q \in V_p. \end{aligned}$$

Uvedený zápis značí, že smerovacia tabuľka uzla p obsahuje všetky uzly a spojenia, s ktorými má uzol p priame spojenie. Záznam príslušných uzlov obsahuje taktiež *pid* a *nid*. V reálnych aplikáciach platí, že dva uzly p' a p'' môžu mať navzájom nekonzistentné informácie, kde $Rp' \subseteq G_i$ a súčasne $Rp'' \subseteq G_{i+n}$, $n > 0$. Samotné smerovanie prebieha podľa algoritmu 1.1.

Algoritmus 1.1 Smerovací algoritmus v modeli P2P siete

if $(p,u) \in E_p$ **then**
Pošli správu uzlu u.
else
Vyber jednu alebo viac prepojení $e = (p,q)$ z E_p na základe smerovacích kritérií a odošli správu uzlu q (v každom kroku je správa bližšie k cieľu).
end if

Ďalej definujeme niektoré pojmy používané v nasledujúcich kapitolách:

Dĺžka cesty – počet prepojení, cez ktoré správa prejde (hops).

Diameter siete – najdlhšia vzdialenosť medzi ľubovoľnými dvoma uzlami siete. Diameter predstavuje najdlhšiu trasu, ktorou môže byť správa v sieti odoslaná (worst-case).

Stupeň uzla – počet susedných uzlov s ktorými má uzol priame prepojenie.

Trasa smerovania – postupnosť uzlov alebo prepojení od zdrojového uzla k cieľovému uzlu správy. Rozlišujeme *rekurzívnu* trasu smerovania a *iteratívnu* trasu smerovania. Pri použití rekurzívnej trasy preposiela každý zúčastnený uzol správu bližšie k cieľu. Pri použití iteratívnej trasy odpovedajú uzly na trase zdrojovému uzlu adresou nasledujúceho uzla. Zdrojový uzol následne znovu odošle správu nasledujúcemu uzlu.

1.2.3 Ukladanie a vyhľadávanie

Väčšina modelov P2P sietí je vhodná pre rôzne oblasti aplikácie. Avšak primárnou aplikáciou je zdieľanie súborov a tomuto účelu je prispôsobená aj použitá terminológia.

V zásade rozoznávame dva prístupy k ukladaniu objektov:

- *Dynamická distribúcia* objektov v sieti.
- *Statické uloženie* objektov v sieti.

Prvý prístup považuje sieť za distribuovanú databázu, kedy nevlastní objekty konkrétny užívateľ, ale sú rovnomerne distribuované v celej sieti na základe určitého algoritmu. Objekty sú následne uložené v oblasti pevného disku, vyhradenej pre tento účel (napr. Freenet). Druhý prístup zachováva vlastníctvo objektov konkrétnymi užívateľmi. V tom prípade je objekt spravidla uložený na pevnom disku užívateľa, ktorý ho zdieľa (napr. Gnutella, eDonkey Network).

Oba prístupy majú svoje klady aj zápory. Pri statickom uložení je obecne jednoduchšie zachovať konzistenciu dát a zaručiť, že nájdený objekt bude možné získať celý aj v prípade, že bol fragmentovaný[4]. Dynamická distribúcia naopak umožňuje získať objekt v sieti aj v prípade, že pôvodný užívateľ, ktorý objekt publikoval, už nie je do siete pripojený. Týmto spôsobom umožňuje dynamická distribúcia prispievať k fungovaniu siete aj užívateľom, ktorí sami žiadny obsah nezdieľajú alebo ho zdieľajú len veľmi málo.

Rozoznávame dva hlavné typy vyhľadávania:

- Vyhľadávanie objektu na základe parametrov objektu. Napr. vyhľadávanie na základe metadát, názvu súboru, obsahu súboru atď. (nejednoznačné).
- Vyhľadávanie objektu na základe identifikátora (jednoznačné).

Tieto dva typy sa v praxi kombinujú, pričom prvý krok je zostavenie zoznamu objektov, ktoré vyhovujú zadaným kritériám a následné vyhľadanie replík týchto objektov v sieti na základe jednoznačného identifikátora. Množinu objektov v sieti značíme S, pričom každý objekt $s \in S$ je zviazaný s uzlom p, v ktorom je uložený (s, p). Identifikátor objektu značíme sid.

V rozsiahlych sieťach je neefektívne prehľadávať celú sieť za účelom nájdenia konkrétneho objektu. Tento problém rieši mechanizmus, ktorý sa nazýva *distribuovaná hashovacia tabuľka* (DHT). Základný princíp spočíva vo vytvorení distribuovaného indexu objektov v sieti, pričom každý uzol je zodpovedný za určitú podmnožinu objektov. V tom prípade stačí smerovať požiadavku na vyhľadanie objektu na podmnožinu uzlov, ktoré sú za tento objekt zodpovedné.

V prípade DHT je priestor identifikátorov objektov rovnaký ako priestor identifikátor uzlov (napr. čísla patriace do rovnakého intervalu). Každý uzol p je zodpovedný za objekty, ktorých sid leží medzi jeho pid a pid jeho nasledovníka q a teda platí $pid_p \leq sid < pid_q$. Priemerný počet objektov, za ktoré p zodpovedá je $|S|/|V|$.

V P2P sieťach používajúcich DHT rozoznávame minimálne dve operácie. Prvá je operácia uloženia objektu $insert(sid, s)$ a druhá operácia získania objektu $s = lookup(sid)$. Smero-

[4]Fragmentácia objektu znamená, že objekt je rozdelený do blokov o pevnej veľkosti (napr. 4 KiB), ktoré je možné vyhľadať a získať samostatne.

vanie v takejto sieti prebieha na základe *kľúča* (key-based routing). Kľúč je rovný hodnote *sid*, na základe ktorej uzol určuje kam požiadavku pri vyhľadávaní prepošle. Požiadavku následne spracuje uzol, ktorého *pid* je najbližšie *sid*.

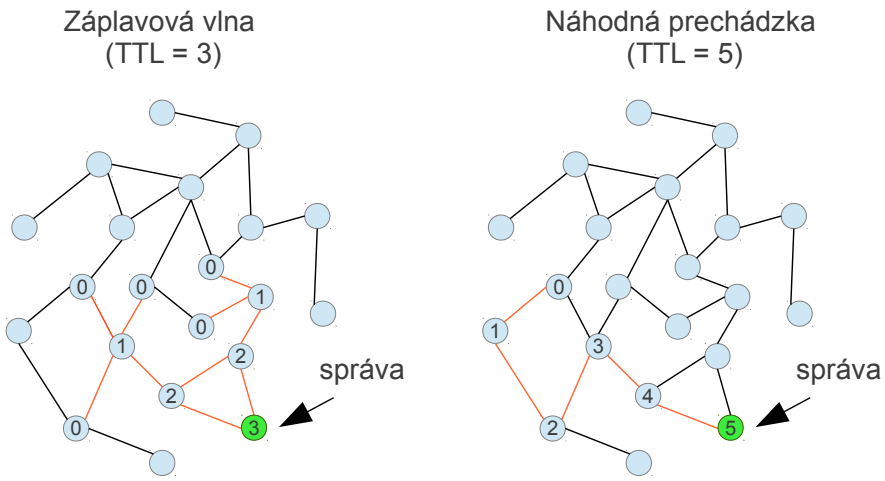
1.3 Neštrukturované siete

V praktických P2P aplikáciach sa najčastejšie stretávame s *neštrukturovanými sieťami*. Jedná sa o siete, ktorých štruktúra je náhodná alebo sú na ňu kladené len minimálne nároky. Príkladom týchto sietí sú Freenet, Gnutella, eDonkey Network, Kazaa, Gossip atď. Neštrukturované siete je jednoduché vytvoriť a udržiavať, ale v prípade rozsiahlejších sietí narážame na problémy s ich efektivitou a problematickým prehľadávaním takejto siete. Existujú techniky, ktoré síce dokážu prehľadávať ľubovoľne veľkú neštrukturovanú sieť, ale množstvo signalizačných správ rastie spolu s veľkosťou siete a pri určitej frekvencii zasielania správ začne efektivita siete klesať [18]. Túto hodnotu nazývame *bod zlomu* (collapse point).

Rozoznávame dve základné metódy smerovania v neštrukturovaných P2P sietiach:

- Smerovanie typu *záplavová vlna* (flooding). Jedná sa o princíp, kedy uzol, ktorý správu prijal odošle správu ďalej všetkým susedným uzlom a dekrementuje hodnotu TTL[5] o jedna. Ak je $TTL = 0$, správa stráca platnosť a ďalej sa sieťou nešíri.
- Smerovanie typu *náhodná prechádzka* (random walk). V tomto prípade je hodnota TTL nastavená na výrazne vyššiu hodnotu, pričom správu uzol neposiela všetkým susedným uzlom, ale iba jednému náhodnému. Rovnako platí, že hodnota TTL je dekrementovaná pri každom prechode uzlom a končí v prípade, že $TTL = 0$.

Oba princípy sú neefektívne vo veľkých sieťach a preto sa v praxi často používajú vylepšené varianty prispôsobené použitej štruktúre P2P siete. Ilustráciu oboch metód môžeme vidieť na obr. 1.4.



Obr. 1.4: Metódy smerovania záplavová vlna a náhodná prechádzka

[5]Time To Live - celé číslo ustanovujúce, koľko uzlov ešte môže správu preposlať, kým skončí jej platnosť.

1.4 Štrukturované siete

Motiváciou k vzniku štrukturovaných sietí je zvýšenie efektivity neštrukturovaných sietí. Prevažná väčšina štrukturovaných sietí k tomuto účelu zavádza koncept DHT predstavený v sekcii 1.2.3. Hlavným prínosom zavedenia DHT je, že vďaka nemu je možné dosiahnuť zložitosť $O(\log N)$, kde N je počet uzlov v sieti. Vyhľadávanie v sieti je teda ekvivalentné binárnemu vyhľadávaniu.

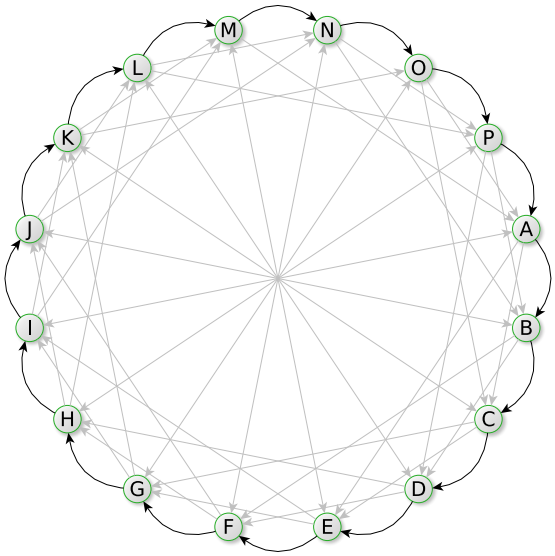
Napriek tomu, že štrukturované siete majú výrazne lepšie teoretické vlastnosti, ich hlavný nedostatok je náročná údržba siete. V prípade, že je priemerná doba zotrvania užívateľa v sieti krátka, môže údržba siete znížiť celkovú efektivitu pod úroveň neštrukturovaných sietí [21]. Základné vlastnosti štruktúry vybraných sietí môžeme nájsť v Tabuľke 1.1.

Medzi najvýznamnejšie štrukturované siete patria Chord, Pastry, Tapestry a Kademlia. Pretože mnoho z nich má podobné vlastnosti, predstavíme si iba niektoré z nich.

1.4.1 Chord

Chord je sieť a zároveň protokol používajúci DHT, pričom patrí k typickým reprezentantom štrukturovaných sietí. Bol vytvorený v roku 2001 na MIT za účelom vytvorenia efektívneho protokolu pre univerzálne použitie [20]. Základný princíp predpokladá použitie konzistentného hashovacieho algoritmu (napr SHA-1) k vytvoreniu jedinečných identifikátorov uzlov a objektov spadajúcich do rovnakého priestoru identifikátorov.

Uzly sú reprezentované kruhom o maximálnej veľkosti 2^m kde m je dĺžka hashu v bitoch. Pre identifikátor pid potom platí $pid \in \{0, 1, 2, \ldots, 2^m - 1\}$. Každý uzol p obsahuje tabuľku ukazovateľov o veľkosti m, kde položka na pozícii i obsahuje ukazovateľ na uzol $(pid + 2^{i-1}) \bmod 2^m$. Táto vlastnosť umožňuje vyhľadávanie v sieti s logaritmickou zložitosťou (viď obr. 1.5).



Obr. 1.5: Štruktúra siete Chord

1.4.2 Pastry

Fungovanie siete Pastry je veľmi podobné Chordu, s ktorým zdieľa základné princípy smerovania a ukladania objektov. Na rozdiel od Chordu zavádza niektoré nové koncepty, medzi ktoré patrí napr. použitie *smerovacích metrík*[6]. Smerovaciu metriku je možné zmeniť alebo upraviť bez zmeny fungovania siete [18].

Medzi ďalšie výhody siete Pastry patrí dobrá odolnosť voči výpadkom uzlov v sieti a úplná decentralizácia. Pridanie nových uzlov do siete je taktiež jednoduché, ale vyžaduje znalosť aspoň jedného uzla existujúcej siete. Vďaka univerzálnej koncepcii tejto siete je možné vytvárať aplikácie využívajúce Pastry protokol s rôznym účelom. Medzi takéto aplikácie paria napríklad distribuovaný súborový systém PAST alebo publikačný systém SCRIBE.

1.4.3 Kademlia

Základnou odlišnosťou vrstvy prepojení Kademlia je oproti Chordu použitie metriky založenej na operácii XOR. Táto metrika $d(x, y)$ v priestore identifikátorov *pid* spĺňa nasledujúce podmienky:

1. $d(x, y) = 0 \Leftrightarrow x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, z) \leq d(x, y) + d(y, z)$

To okrem iného umožňuje efektívne smerovanie, ale pretože operácia XOR formuje ábelovsku grupu, umožnuje P2P sieť Kademlia taktiež jednoduchšiu formálnu analýzu. Medzi ďalšie nové prvky, ktoré Kademlia priniesla, patria paralelné vyhľadávacie dotazy [5].

1.4.4 Základné vlastnosti

Na záver tejto sekcie si zhrnieme základné vlastnosti najpoužívanejších štrukturovaných sietí vo forme tabuľky (viď tab. 1.1) [5].

Typ	**Stupeň uzla**	**Diameter**
Chord	$\log_2(N)$	$\log_2(N)$
CAN	$2d$	$\frac{1}{2}\, dN\, \frac{1}{d}$
Pastry	$(b-1)\, \log_b(N)$	$\log_b(N)$
Kademlia	$k \log_2(N)$	$\log_2(N)$
Trie	$k+1$	$2\, \log_k(N)$
de Bruijin	k	$\log_k(N)$

Tabuľka 1.1: Zhrnutie základných vlastností štrukturovaných sietí

[6] Medzi typické smerovacie metriky patrí napríklad počet sieťových prvkov na ceste k cieľovému uzlu, čas odozvy (RTT), geografická poloha atď.

Kapitola 2

Anonymita v P2P sieťach

P2P siete majú za sebou dlhú históriu spojenú so vznikom viacerých pohľadov a názorov na ich fungovanie. Počas tejto doby sa okrem technických inovácii v oblasti IT (napríklad rýchlosť internetu a jeho dostupnosť) menili aj nároky ich používateľov. Jednou z požiadaviek v špecifických aplikáciach je anonymita užívateľov v sieti, ktorej mieru vieme garantovať na základe konkrétnych matematických modelov zabezpečujúcich anonymnú komunikáciu medzi dvoma alebo viacerými entitami v sieti.

Aplikácie, v ktorých nájdu anonymizačné techniky uplatnenie sú napríklad elektronické hlasovanie v rámci určitej organizácie alebo štátu, necenzurovateľné publikačné systémy, anonymná textová alebo hlasová komunikácia atď. Aplikácie ako Tor, Freenet alebo GNUnet sú často používané za účelom obchádzania štátnych regulácií a cenzúry internetu [14].

Definícia 2.0.1. *Predpokladajme zaslanie správy m z uzla a (autor) do uzla b (príjemca). Za úplne anonymnú budeme P2P sieť považovať vtedy, ak a ani b nie je schopný určiť identitu toho druhého s pravdepodobnosťou:*

$$p > N^{-1}$$

kde N je počet uzlov v sieti.

Definícia 2.0.2. *Predpokladajme zaslanie správy m z uzla a (autor) do uzla b (príjemca). Za čiastočne anonymnú budeme P2P sieť považovať vtedy, ak a alebo b je schopný určiť príslušnosť toho druhého do podmnožiny W všetkých uzlov V, ale nie je schopný určiť, o ktorý uzol podmnožiny sa jedná s pravdepodobnosťou:*

$$p > \frac{1}{|W|}$$

Pre hodnotenie miery anonymity sa používa viacero spôsobov [18]. Prvý spôsob je obrátenou hodnotou pravdepodobnosti p_x, ktorá udáva pravdepodobnosť, že určitý uzol x je autorom správy m a teda pre mieru anonymity d_x platí:

$$d_x = 1 - p_x \tag{2.1}$$

Druhý používaný spôsob hodnotí mieru anonymity globálne pre súbor pravdepodobností p pre všetky uzly v sieti na základe entropie, kde X je diskrétna náhodná veličina s distribučnou funkciou $p_x = P(X = i)$ kde x reprezentuje všetky hodnoty, ktoré môže premenná X nadobudnúť s pravdepodobnosťou $p_x > 0$. V anonymných sietiach x reprezentuje uzol v sieti a p_x pravdepodobnosť, že x je autorom správy. Entropiu potom vypočítame ako:

$$H(X) = -\sum_{x=0}^{N-1} p_x \log_2(p_x) \tag{2.2}$$

Miera anonymity na základe entropie sa udáva v normalizovanej forme kde $d \in \langle 0, 1 \rangle$. Normalizujeme cez maximálnu hodnotu entropie H_M a teda:

$$d = \frac{H(X)}{H_M} \tag{2.3}$$

V tejto práci si však vystačíme s prvou uvedenou mierou d_x.

Anonymné siete úzko súvisia s *necenzúrovateľnými sieťami* (censorship resistant networks), tie spolu tvoria sieť, v ktorej nie je možné identifikovať autora, ktorý objekt publikoval a zároveň nie je možné technicky vynútiť odstránenie tohoto objektu zo siete. Tento mechanizmus je možné rozšíriť o hodnotenie objektov (recenzie), vďaka ktorému môžu byť nechcené objekty zo siete odstránené na základe kolektívneho rozhodnutia užívateľov systému.

Súčasné anonymizačné techniky využívajú jeden alebo viac z troch základných mechanizmov [18]:

Source-rewriting – princíp spočíva v smerovaní správ cez viacero náhodných uzlov, pričom príjemca správy nemá možnosť zistiť či je odosielateľ správy skutočný zdroj správy alebo správu iba preposiela.

Broadcast protocols – skupina protokolov využívajúca broadcast[1] pre väčšie skupiny užívateľov, pričom skutočným príjemcom správy je iba jeden. Podobne ako v predchádzajúcom prípade, nie je možné určiť kto je skutočný príjemca správy. Samotný broadcast musí byť implementovaný na aplikačnej úrovni.

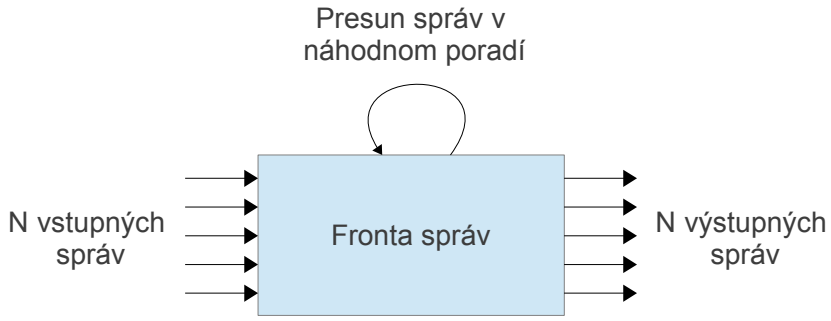
Dining-Cryptographers – použitie problému Dining-Cryptographers, ktorý navrhol v roku 1988 David Chaum. Ten umožňuje anonymné získanie jednobitovej správy v trojčlennej skupine účastníkov. Tento problém je možné následne rozšíriť na ľubovoľný počet účastníkov a ľubovoľne veľkú správu.

2.1 Mix-net

Smerovací protokol Mix-net predstavil v roku 1981 David Chaum [6]. Jeho cieľom je vytvoriť komunikáciu, ktorú je ťažké sledovať a identifikovať pôvodcu správy. K tomuto účelu používa proxy servery, v pôvodnej terminológii nazývané *mixes*. Tie obsahujú frontu správ z rôznych zdrojov, tie následne pomiešajú a prepošlú v inom poradí ako prišli (viď obr. 2.1).

[1] Typ prenosu správy, kedy správu príjmu všetci užívatelia siete.

Samotný protokol funguje tak, že odosielateľ A chce poslať anonymne správu m príjemcovi B. Odosielateľ A vytvorí správu $m' = E_{k_x}(n_1, E_{k_b}(n_0, m), B)$ a odošle ju mixu x. Kde E je dohodnutý algoritmus šifrovania pomocou verejného kľúča k_x mixu x. Ten správu dešifruje a získa tak správu $(n_1, E_{k_b}(n_0, m), B)$. Mix ďalej prepošle správu $m'' = E_{k_b}(n_0, m)$ skutočnému príjemcovi B. Keďže správa je šifrovaná verejným kľúčom príjemcu, nikto okrem A a B nemá prístup k nešifrovanému obsahu m. Hodnoty n_i sú náhodné čísla[2], ktoré zabraňujú identifikácii správ s nešifrovaným obsahom, ktorý už bol použitý v predchádzajúcej výmene správ. Správa môže byť smerovaná cez jeden alebo viacero mixov.



Obr. 2.1: Architektúra smerovacieho protokolu Mix-net

Mix-net poskytuje vysokú mieru anonymity aj v prípade, že útočník môže zachytiť všetky správy prenášané v sieti. Takéhoto útočníka budeme nazývať *globálny útočník*, pričom druhým typom je *lokálny útočník*, ktorý má prístup iba k obmedzenej časti siete. Cenou za túto vlastnosť je vysoká latencia, ktorá vzniká pri čakaní správ vo fronte (fronta musí byť plná, aby mohlo byť zmenené poradie správ). Pôvodne bol protokol navrhnutý pre emailovú komunikáciu, kde je mierna latencia prijateľná. Príkladom anonymných sietí založených na Mix-nete sú GNUnet a Mixmaster.

V súčasnosti sa podobný princíp uplatňuje napríklad v sieti Bitcoin. Princíp návrhu tejto kryptomeny zaručuje, že každý užívateľ môže zistiť zostatok v peňaženke iného užívateľa. Za účelom anonymizácie obsahu peňaženky alebo určitej transakcie, je možné nedeterministicky pomiešať viacero platieb v jednej peňaženke a následne finančné prostriedky previesť do nových peňaženiek. Po vykonaní takejto operácie nie je možné určiť komu peniaze v nových peňaženkách patria. Takáto služba sa nazýva mix service.

2.2 Onion routing

Onion routing je technika predstavená Davidom Goldschlagom, Michaelom Reedom a Paulom Syversonom v roku 1998 [15]. Podobne ako iné anonymizačné techniky umožňuje anonymnú komunikáciu prostredníctvom počítačovej siete, pričom je chránená identita autora správy i jej príjemcu. K tomuto účelu využíva uzly nazývané *smerovače* (onion routers). Správy sú šifrované vo viacerých vrstvách, z čoho je odvodený aj samotný názov.

Predpokladajme, že odosielateľ A chce odoslať správu príjemcovi B. V tom prípade A náhodne zvolí niekoľko smerovačov v sieti, cez ktoré bude správa prenášaná. Nazvime

[2] V kryptografii označované ako *nonce*.

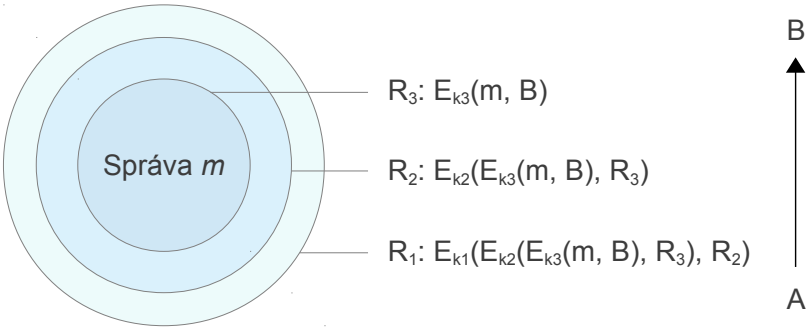
ich $R_1, R_2, \ldots, R_n$, pričom R_i má verejný kľúč k_i. Potom odosielateľ A odošle správu $m' = E_{k_1}(E_{k_2}(\ldots(E_{k_n}(m, B), R_n)\ldots, R_3), R_2)$ smerovaču R_1, ten správu dešifruje svojim privátnym kľúčom a ďalej pošle správu $m'' = E_{k_2}(\ldots(E_{k_n}(m, B), R_n)\ldots, R_3)$ smerovaču R_2. Posledný uzol R_n odošle správu príjemcovi B. Odpoveď príjemcu B bude odoslaná rovnakou trasou v reverznom poradí.

Je dôležité si uvedomiť, že posledný uzol na trase vidí nešifrovaný obsah správy m. V praktických aplikáciach toto nie je problém, ak je použité šifrovanie na transportnej alebo aplikačnej vrstve (napríklad TLS). Onion routing taktiež nie je bezpečný v prípade globálneho útočníka, pretože medzi odoslanými a prijatými správami jednotlivých uzlov existuje časová korelácia, ktorú je možné využiť pri časovej analýze prenášaných správ a identifikovať tak dvojice odosielateľ-príjemca (A, B).

Najznámejšou implementáciou onion routingu je Tor[3], ktorý prináša niekoľko vylepšení protokolu. Výrazný rozdiel je v spôsobe výberu použitých smerovačov. Tor používa pri výbere nasledujúce metriky:

- Smerovač je označený ako *stabilný*, ak je jeho dostupnosť lepšia ako medián dostupnosti všetkých smerovačov v sieti.
- Smerovač je označený ako *rýchly*, ak je jeho prenosová rýchlosť vyššia ako medián prenosových rýchlostí všetkých smerovačov v sieti.

Vstupný smerovač musí byť označený ako stabilný a zároveň rýchly. Ostatné smerovače sú vyberané tak, aby bol prenos čo najoptimálnejší. Medzi ďalšie rozdiely patrí spôsob vytvárania trasy prenosu, kedy Tor používa inkrementálne vytváranie prenosovej cesty. To znamená, že odosielateľ A najprv vytvorí TCP tunel so smerovačom R_1, prostredníctvom tohoto tunelu vyjedná spojenie s R_2, atď až k príjemcovi B. Zároveň je počas vytvárania trasy vygenerovaný unikátny kľúč relácie pre každé spojenie. Tieto kľúče sú následne použité pri odoslaní odpovede odosielateľovi A. Tor používa 3 smerovače pre prenos správy (viď obr. 2.2).



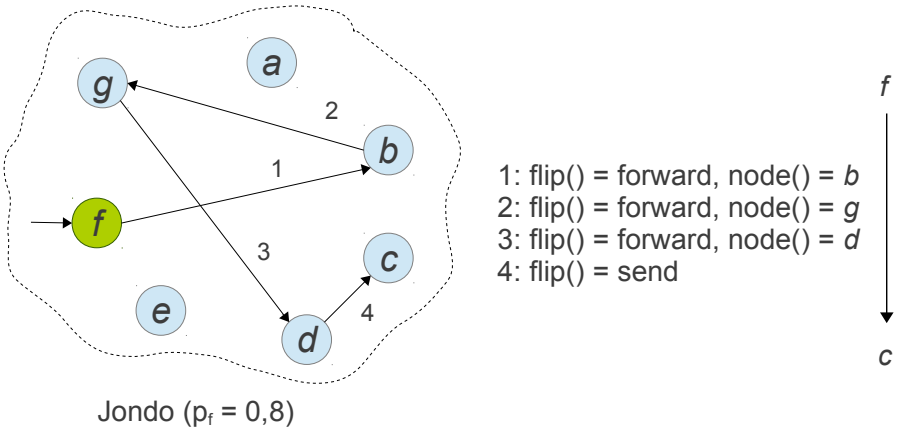
Obr. 2.2: Zapuzdrenie jednotlivých správ v protokole aplikácie Tor

[3]Systém určený na anonymné prehliadanie webu.

2.3 Crowds

Crowds je protokol určený pre webové transakcie navrhnutý Michaelom K. Reiterom a Avielom D. Rubinom v roku 1998 [16]. Protokol používa skupiny uzlov označované ako *crowds*, ktoré môžu komunikovať s webovým serverom anonymne. Základná myšlienka spočíva v tom, že jednotlivé uzly preposielajú požiadavky medzi členmi skupiny, pričom každý člen sa náhodne rozhoduje s pravdepodobnosťou p_f, či pošle požiadavku priamo serveru alebo použije ďalšieho člena ako proxy server. Členovia skupiny sú chápaní ako bežiace procesy označované ako *jondo* (z John Doe). Komunikácia prebieha nasledovne:

1. Nový člen sa registruje do skupiny prostredníctvom serveru označovaného ako *blender*. Ten je zodpovedný za správu členov skupiny, notifikáciu v skupine o príchode nového člena a distribúciu kľúčov.
2. Každý člen, ktorý chce odoslať novú požiadavku si „hodí mincou“, na ktorej padá hlava s pravdepodobnosťou $p_f > 0,5$. Ak padne hlava, prepošle požiadavku na iného náhodne zvoleného člena skupiny, inak odošle požiadavku priamo na cieľový server. Je teda možné, že autor požiadavky ju pošle priamo na server alebo sám sebe. Ak je požiadavka preposlaná na ďalšieho člena skupiny, hod mincou sa opakuje (viď Obr 2.3).
3. Ak je člen skupiny vyzvaný k spracovaniu požiadavky, je povinný odstrániť z požiadavky informácie o jej autorovi.
4. Žiadny člen skupiny nemá možnosť s určitosťou zistiť, či požiadavka pochádza priamo od jej autora alebo nastalo preposlanie.
5. Odpoveď je odoslaná zdroju rovnakou cestou, ktorá vznikla spôsobom popísaným v bode 2.



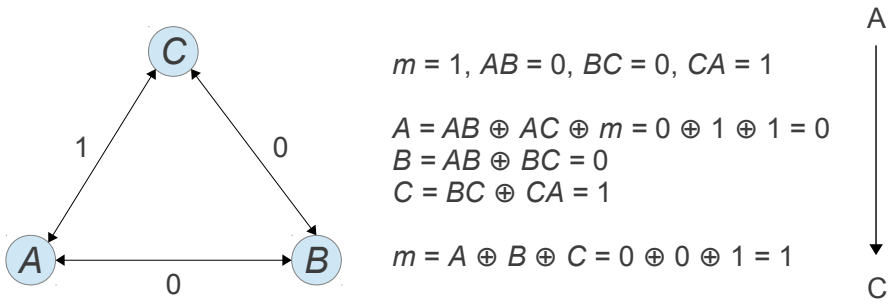
Obr. 2.3: Preposielanie správ v anonymizačnej sieti Crowds

Crowds poskytuje dobrú mieru anonymity, ktorá stúpa spolu s počtom uzlov N. Existuje však viacero útokov, ktoré jej použitie v určitých prípadoch obmedzujú [22]. Taktiež nie je možné zaručiť bezpečnosť v prípade globálneho útočníka alebo väčšieho počtu spolupracujúcich uzlov.

2.4 DC-Net

Anonymizačná sieť DC-Net využíva *problém večerajúcich kryptografov* (Dining Cryptographers) [7] k zaisteniu úplnej anonymity medzi N účastníkmi. Základný model pracuje len s tromi. Komunikačný protokol je založený na použití operácie XOR ku kombinácií odpovedí všetkých zúčastnených, pričom súčasne môže byť autorom správy iba jeden z nich. DC-Net funguje nasledovne:

1. Predpokladajme troch účastníkov komunikácie: Alica (A), Bob (B) a Cyril (C). Alica chce poslať správu Cyrilovi (viď obr. 2.4).
2. Každá dvojica zúčastnených uskutoční tajný hod mincou, ktorého výsledok ostatní účastníci nevedia. Výsledky tohoto hodu nazvime AB, BC, CA, kde jedna strana mince reprezentuje hodnotu 0 a druhá 1.
3. Bob a Cyril zverejnia výsledky svojich dvoch hodov, kombinované pomocou operácie XOR, teda $B = AB \oplus BC$ a $C = BC \oplus CA$.
4. Alica chce odoslať jednobitovú správu m a to takým spôsobom, že ju pripojí ku kombinácii svojich dvoch hodov, teda $A = AB \oplus CA \oplus m$.
5. Pretože sa nachádza výsledok každého hodu vo všetkých správach celkovo dvakrát, môžeme pomocou operácie XOR získať správu ako $m = A \oplus B \oplus C$, pretože platí $x \oplus (y \oplus y) = x \oplus 0 = x$. Túto správu vidia všetci účastníci komunikácie (vrátane potenciálneho útočníka), nie je však možné zistiť, kto je autorom správy. V tomto prípade sa taktiež predpokladá, že nemôže dôjsť k súčasnému odoslaniu viac ako jednej správy.



Obr. 2.4: Komunikačný protokol DC-Net

Podobne ako u mnoho iných systémov využívajúcich broadcast, klesá šírka pásma kvadratický s počtom zúčastnených. Preto sa častejšie používajú upravené varianty, ktoré rozdeľujú účastníkov na menšie skupiny s cieľom zvýšiť efektivitu siete (napr. Herbovire). Okrem tohoto obmedzenia patrí medzi nevýhody taktiež:

- Protokol samotný predpokladá existenciu bezpečných komunikačných kanálov medzi účastníkmi, čo v praxi nie je vždy bezproblémové. Najčastejšie je riešením použitie asymetrickej kryptografie.
- Predpoklad bezkolízneho odoslania správy často nie je zlúčiteľný s reálnymi aplikáciami a preto je nutné implementovať vzájomne výlučné odosielanie správ podľa

určitého harmonogramu, systému priorít, rezervácie slotov atď.

- Integritu správy je možné jednoducho narušiť hociktorým účastníkom komunikácie a odoslanie správy tak zlyhá. Tomuto nie je možné predchádzať, avšak vieme túto situáciu detektovať. V systémoch založených na viacerých oddelených skupinách účastníkov je možné skupinu obsahujúcu narušiteľa opustiť a začleniť sa do inej.

2.5 Zhrnutie a porovnanie anonymizačných techník

Všetky uvedené techniky si na záver zhrnieme vo forme tabuľky (viď tab. 2.1). Tabuľka obsahuje základné vlastnosti ovplyvňujúce použiteľnosť v P2P sieťach. Všetky parametre sa vzťahujú na pôvodné techniky. Existujú modifikované verzie, ktoré majú lepšiu škálovateľnosť (predovšetkým Herbovire).

Technika	**Mix-net**	**Onion routing**	**Crowds**	**DC-Net**
Odolnosť voči odposluchu[4]	globálny útočník	lokálny útočník	lokálny útočník	globálny útočník
Latencia	vysoká	nízka	nízka	nízka
Škálovateľnosť	vysoká	vysoká	stredná	nízka
Počet medziuzlov[5]	konštantný	konštantný	$< 0, N >$	teoreticky N
Konektivita	-	-	N	N

Tabuľka 2.1: Zhrnutie a porovnanie anonymizačných techník

[4] Určuje nutné možnosti útočníka, aby bol schopný priradiť zdroj správy k jej cieľu.
[5] Počet uzlov ležiacich medzi zdrojom a príjemcom správy

Kapitola 3

Bezpečnosť P2P sietí

Komunikačné a softwarové systémy, medzi ktoré patria aj P2P siete hodnotíme na základe viacerých parametrov. Medzi najdôležitejšie parametre patrí jednoznačne bezpečnosť. Užívateľ P2P siete vyžaduje predovšetkým ochranu citlivých dát a osobných údajov, spoľahlivosť prevádzky a dôveru k poskytovaným informáciám. Obecne platí, že v centralizovaných systémoch máme lepšiu kontrolu nad jeho užívateľmi a dátami a nie je tomu inak ani v P2P sieťach, kde je bezpečnosť distribuovaných systémov stále otvoreným problémom. Útoky môžeme rozdeliť podla nasledujúcej klasifikácie [18]:

Podľa motivácie útočníka:

- *Náhodné útoky* (chyba SW) – kategória útokov, kedy môže k narušeniu bezpečnosti dôjsť i pri bežnom používaní klientského softwaru v dôsledku závažnej chyby v návrhu aplikácie.
- *Cielené útoky* – v tomto prípade nie je možné narušiť bezpečnosť pri bežnej práci s klientským softwarom, ale je potrebné použiť techniky reverzného inžinierstva, manipuláciu s paketmi na nižších vrstvách, atď.

Podľa výsledku útoku:

- *Porušenie autorizácie* – útočník je schopný vykonať operáciu ku ktorej by nemal byť za bežných okolností oprávnený (zmazanie objektu, získanie objektu atď.).
- *Porušenie autentifikácie* – útočník je schopný obísť proces autentifikácie a môže sa tak vydávať za iného užívateľa v sieti.
- *Porušenie dôvernosti* – útočník môže pristupovať k informáciám, ktoré sú určené pre iného užívateľa v sieti alebo by nemali byť prístupné vôbec.
- *Porušenie integrity* – útočník môže modifikovať dáta, ktorých autorom nie je on sám.

Podľa dopadu na obeť útoku:

- *Prerušenie správnej funkcie SW* – po útoku dôjde k porušeniu funkcie softwaru, ale nie je pri tom porušená bezpečnosť dát a zároveň je možné obnoviť pôvodnu funkčnosť.
- *Neautorizovaný prístup* – po útoku môže útočník získať prístup k chránenému obsahu.

- *Neautorizovaná modifikácia* – po útoku môže útočník modifikovať obsah, ktorý mu nepatrí a nie jeho autorom.
- *Neautorizované vytvorenie* – po útoku môže útočník vytvoriť obsah v mene iného používateľa klientského SW.

Slabým miestom každej centralizovanej siete je práve jej centrálny server alebo skupina serverov. Mnoho novovznikajúcich P2P sietí sa snaží vyhnúť použitiu centralizovaných riešení, práve z dôvodu závislosti celej siete na bezpečnosti jedného miesta v sieti.

3.1 Typické útoky

Každý komunikačný system má jedinečné vlastnosti, architektúru a implementáciu. Vďaka tomu sa môžu v každom systéme vyskytnúť špecifické útoky, ktoré nemusia byť na alternatívne systémy aplikovateľné. Napriek tomu pozorujeme v P2P systémoch určité vzory útokov, ktoré sa často vyskytujú na rôznych vrstvách (vrstva sietových prepojení, aplikačná alebo sieťová). Na tieto útoky [5][3][18] sa pozrieme bližšie v tejto sekcii.

3.1.1 Útoky na nodeID

Tento útok je typický pre štrukturované siete, ktoré k svojej funkcii používajú v rôznych podobách identifikátor uzla známy ako *nodeID*. Podstata tohoto útoku spočíva v možnosti klientov voliť si hodnotu *nodeID* samostatne, bez nutnosti požiadať o vytvorenie dôveryhodnú tretiu stranu. V tom prípade môže útočník generovať *nodeID* tak, aby všetky jeho uzly obsadili špecifické pozície v sieti (napr. aby zodpovedali za všetky kópie jedného objektu). Dopad takéhoto útoku môže byť získanie plnej kontroly nad určitým objektom a následné cenzurovanie tohoto objektu, jeho vymazanie alebo modifikácia.

Obrana

Typická obrana spočíva vo vytvorení dôveryhodnej centralizovanej autority, ktorá zodpovedá za bezpečné prideľovanie identifikátorov a príslušných klientských certifikátov (overenie identifikátoru následne zabezpečuje asymetrická kryptografia).

Toto riešenie však nie je možné uplatniť vo všetkých typoch P2P sietí a preto sa hľadajú vhodné decentralizované riešenia. Medzi takéto riešenia patrí decentralizovaná evidencia použitých IP adries, kedy každý uzol obsahuje zoznam dvojíc (*nodeID*, *IPadresa*). V tom prípade je však potrebné kontrolovať hodnoty deklarované ostatnými uzlami, aby sme predchádzali ukladaniu falošných záznamov. To by otvorilo možnosti DoS útoku, kedy by útočník mohol zablokovať prístup legitímnym užívateľom. Ďalšou nevýhodou tohoto riešenia, je diskriminácia uzlov používajúcich NAT[1], ktoré zdieľajú rovnakú IP adresu.

V neposlednom rade môžeme zaviesť distribuované vytvorenie identity uzla, kedy je užívateľ systému povinný na každú registráciu vynaložiť určité úsilie alebo zdroje (CAPTCHA,

[1] Network Address Translation

poplatok za registráciu, Proof-of-work[2] atď.). Na túto problematiku sa bližšie pozrieme v kapitole 5.

3.1.2 Útoky na smerovacie tabuľky

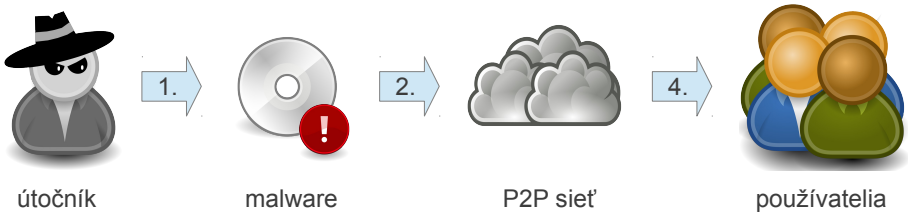
Jednotlivé uzly za účelom optimalizácie spolupracujú a vymieňajú si navzájom informácie o štruktúre siete, z ktorých si následne vytvárajú smerovacie tabuľky. Propagácia nepresných alebo zavádzajúcich informácií v sieti môže spôsobiť chyby v smerovacích informáciách a následne zhorší efektivitu smerovania, zmení trasy smerovania, prípadne zamedzí posielaniu správ úplne. Typickým dôsledkom týchto útokov je presmerovanie správ obete do destinácie zvolenej útočníkom alebo zamedzenie niektorým operáciám v sieti (DoS na strane užívateľa siete).

Obrana

Obrana voči útokom na smerovacie tabuľky pozostáva z dvoch línií obrany. Prvá línia obrany je podobná ako v prípade útokov na *nodeID* a spočíva v bezpečnom prideľovaní identifikátorov, pretože tým sťažíme zacielenie útokov na konkrétne uzly. Druhá línia spočíva v použití bezpečného smerovanie, kedy je správa súčasne zasielaná rôznymi cestami. V takom prípade zvýšime pravdepodobnosť doručenia aj v prípade, že na trase ležia uzly útočníka.

3.1.3 Otrávenie objektu

Spôsob útoku založený na podstrčení škodlivého objektu je typický predovšetkým pre aplikácie na zdieľanie súborov. Cieľom útočníka je najčastejšie rozšíriť v P2P sieti malware[3]. Útočník sa týmto počínaním najčastejšie snaží získať finančné prostriedky buď priamo z účtov napadnutých používateľov alebo nepriamo prenajímaním vytvoreného botnetu[4], príjmu z reklamy atď.



Obr. 3.1: Šírenie malwaru v P2P sieti

[2]Opatrenie, ktoré vyžaduje na potvrdenie určitej operácie vynaloženie určitého množstva práce (najčastejšie výpočty na CPU).

[3]Škodlivý software určený k narušeniu funkcie počítača, zbieraniu citlivých informácií alebo na získanie prístupu k napadnutému systému.

[4]Sieť napadnutých počítačov, ktorú útočník využíva k rozosielaniu nevyžiadanej pošty alebo k DDoS útokom.

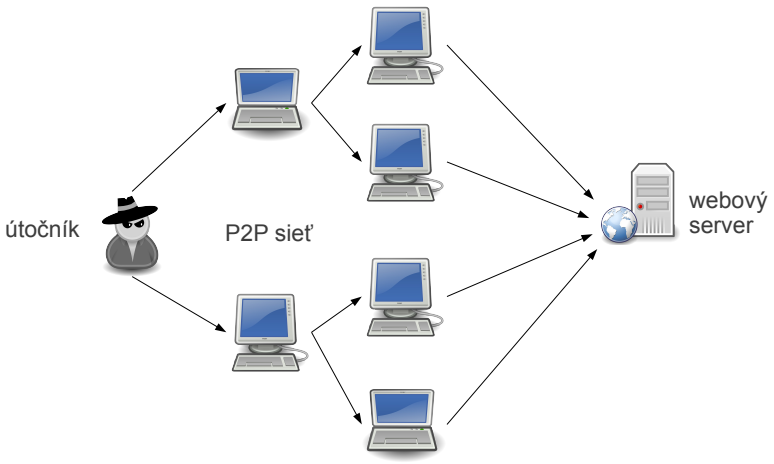
Obrana

Obrana voči malwaru presahuje rozsah tejto práce. V prostredí P2P sietí je minimálne vhodné spoľahlivo identifikovať autorov publikovaných súborov, čo zníži odhodlanie útočníkov. Ak to nie je možné (napr. v prostredí pseudoanonymných sietí) môžeme taktiež použiť algoritmy založené na reputácií užívateľov, ktoré počet útokov znížia.

3.1.4 DoS útoky

V prípade DoS útokov musíme rozlišovať dve varianty prevedenia. V prvej variante je cieľom útoku zariadenie, na ktorom beží P2P software a v druhej je cieľom útoku iné zariadenie pripojené k internetu, ktoré ale nie je súčasťou P2P siete. Tieto útoky majú nasledujúcu charakteristiku:

- *Útok na zariadenie v P2P sieti* – útočník sa snaží zahltiť dostupné systémové prostriedky používateľa pomocou zasielania paketov vyžadujúcich náročné výpočty, sieťovú aktivitu alebo miesto na disku. V niektorých prípadoch je cieľom útočníka otráviť smerovacie tabuľky nesprávnymi informáciami, čim sa smerovanie v sieti stane neefektívne.
- *Útok na zariadenie mimo P2P sieť* – útočník sa pokúša podstrčiť uzlom v sieti nesprávne informácie, ktoré spôsobia odoslanie dotazov na adresy mimo P2P sieť. Zariadenie, na ktoré sú tieto dotazy smerované spravidla nerozozná komunikačný protokol, čo dopad tohoto útoku znižuje (útoky na aplikačnej úrovni sú obtiažne). Napriek tomu môže takýto útok produkovať desiatky tisíc sieťových pripojení z rôznych adries, čo môže byť dostatočné na narušenie funkcie menej odolných systémov. Z hľadiska napadnutého systému sa jedná o distribuovaný DoS útok (DDoS), ktorý môžeme vidieť na obr. 3.2.



Obr. 3.2: Princíp DDos útoku pomocou P2P siete

V oboch prípadoch je riziko o to vyššie, že uzly medzi sebou spolupracujú a chybné informácie v smerovacích tabuľkách alebo DHT poskytnuté útočníkom sa môžu rozšíriť do mnohých ďalších uzlov (potenciálne do celej siete).

Obrana

Existuje mnoho techník obrany voči DoS útokom, pričom väčšina z nich sa v súčastnosti používa aj mimo oblasť P2P sietí. Medzi základné techniky radíme detekciu vzorov útokov (uzol obsahuje databázu signatúr), limitovanie prostriedkov, filtrovanie nežiadúcich uzlov alebo dát a detekciu anomálií.

Proti niektorým typom útokov sa používa aj technika *delayed binding*, ktorá používa ochranný interval pri komunikácii medzi klientom a serverom. Uzol v tomto prípade počká s vytvorením TCP spojenia do doby, keď má dostatok redundantných informácií od ostatných uzlov.

V neposlednom rade je možné použiť aj princíp *Proof-of-work*, kedy uzol požadujúci určitú operáciu musí vykonať približne ekvivalentné množstvo práce, aby mohol požadovať vykonanie tejto operácie iným uzlom.

3.1.5 Sybil útoky

Sybil útoky súvisia s útokmi na *nodeID*. Názov tohoto útoku je pomenovaný podľa rovnomennej knihy, kde vystupuje žena s duševnou chorobou zahŕňajúcou viacero identít. Samotný útok spočíva práve v používaní mnohých identít jedným uzlom alebo úzkou skupinou uzlov. Mnoho identít (hodnôt *nodeID*) umožňuje útočníkovi kontrolovať veľkú časť P2P siete, čo otvára dvere rôznym ďalším útokom.

Obrana

Obrana je podobná ako v prípade útokov na *nodeID*.

3.2 Ostatné útoky

Okrem útokov uvedených v sekcii 3.1 existujú taktiež útoky aplikovateľné len na určitý protokol alebo P2P aplikáciu. Podstata týchto útokov často vyplýva zo špecifického zamerania týchto aplikácií a nie je ich možné zovšeobecniť.

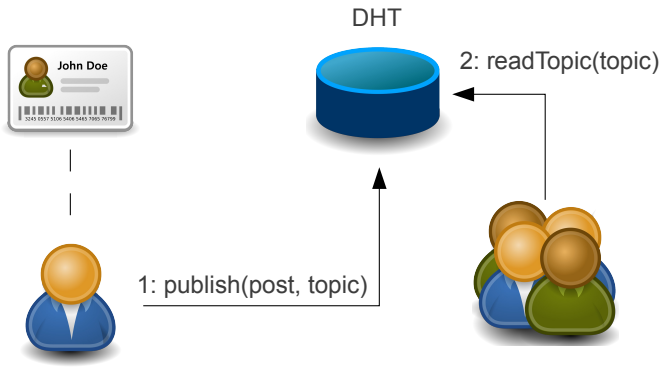
Pre anonymizačné P2P siete existujú taktiež špecifické útoky, ktoré zvyšujú šancu na zistenie identity určitého užívateľa alebo účinnosť anonymizácie degradujú úplne. Týmto útokom sa budeme v kontexte protokolu LSPP podrobnejšie venovať v kapitole 5.

Kapitola 4

Publikačný protokol LSPP

Súčasťou tejto práce je návrh *Liberty Share Publishing Protocol* (LSPP). Úlohou tohoto protokolu bude vytvoriť anonymný necenzúrovateľný publikačný systém, ktorý môže slúžiť pre publikovanie krátkych textových správ alebo ako komunikačný prostriedok medzi jeho užívateľmi. Na protokol LSPP budeme klásť nasledujúce požiadavky:

- Publikovanie v sieti musí poskytovať úplnú alebo čiastočnú anonymitu tak, ako sme si ju definovali v definícii 2.0.1. Ak je táto podmienka splnená, systém zabezpečuje *anonymitu*.
- Obsah príspevku nie je možné upraviť alebo odstrániť zo siete bez zásahu autora príspevku. Táto podmienka môže byť splnená buď úplne (príspevok v sieti existuje až pokiaľ sa neodpojí posledný uzol) alebo čiastočne (príspevok v sieti existuje až pokiaľ sa neodpojí určitý počet uzlov obsahujúcich príspevok samotný). Pokiaľ je táto podmienka splnená, systém označujeme ako *necenzúrovateľný*.
- Identita autora príspevku je daná jeho jednoznačným identifikátorom. Pomocou tohoto identifikátora však nesmie byť možné určiť IP adresu autora alebo jeho skutočnú identitu. Každý identifikátor autora musí byť neduplikovateľný (i.e., nesmie byť možné vydávať sa za niekoho iného).



Obr. 4.1: Koncept fungovania protokolu LSPP

Samotná koncepcia systému je podobná aplikácii Twitter[1]. V sieti je teda možné vykonávať nasledujúce operácie:

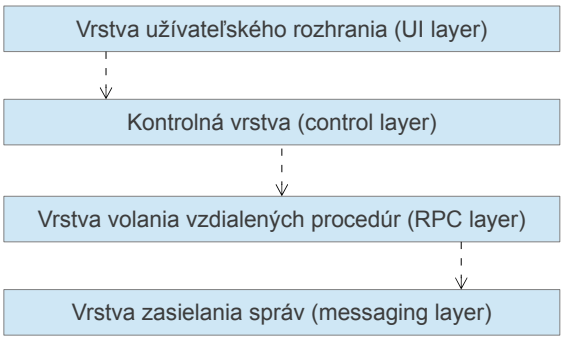
- Autor príspevku (post) môže publikovať obsah pod svojim pseudonymom (ID) a následne k tomuto obsahu môže pristupovať každý, kto je odberateľom príspevkov autora.
- Autor môže publikovať príspevok týkajúci sa určitej témy (topic). K tomuto obsahu môže pristupovať každý, kto je čitateľom danej témy.
- Užívateľ môže adresovať príspevok konkrétnemu užívateľovi. V tomto prípade nie je schopný správu prečítať nikto iný, ako adresát príspevku.

4.1 Architektúra

Po definovaní požiadaviek na fungovanie publikačného systému Liberty Share, je nutné implementovať P2P sieť, ktorá sa vyznačuje uvedenými vlastnosťami. Naším cieľom bude navrhnúť protokol, ktorý je bezpečný a odolný voči širokej škále útokov, ale zároveň dostatočne efektívny a vhodný pre praktické použitie.

Prvým krokom je výber medzi štrukturovaným a neštrukturovaným usporiadaním siete. Ako sme už načrtli v sekcii 1.3, hlavným nedostatkom neštrukturovaných sietí je ich neschopnosť spoľahlivo zabezpečiť nájdenie objektu v sieti. Na druhej strane, štrukturované siete poskytujú viac informácií o vzájomných prepojeniach a sú zároveň náchylnejšie na útoky, ktoré túto vlastnosť využívajú. Pre protokol LSPP bolo zvolené štrukturované usporiadanie typu Chord, rozšírene o dodatočné bezpečnostné mechanizmy. To zabezpečí spoľahlivé publikovanie príspevku s garantovanou dĺžkou cesty a teda aj latenciou.

Architektúra publikačného systému je rozdelená do vrstiev (viď obr. 4.2). Pričom každá nadradená (vyššie položená) vrstva poskytuje vyššiu mieru abstrakcie. Prerušované šípky značia závislosť tak ako ju poznáme z jazyka UML. Implementačné zmeny vo vyššej vrstve preto neovplyvnia nižšie vrstvy. Implementačné zmeny v nižšej vrstve ovplyvnia jej nadradenú vrstvu, ale už nie vyššie položené vrstvy.



Obr. 4.2: Vrstvy architektúry protokolu LSPP

[1]Twitter je mikroblogovacia služba (taktiež sociálna sieť), ktorá umožňujúce užívateľom publikovať a čítať krátke textové správy.

Jednotlivé vrstvy zabezpečujú nasledujúce funkcie:

Vrstva zasielania správ – jedná sa o základnú vrstvu, ktorá zabezpečuje komunikáciu medzi uzlami v sieti. Toto je zároveň jediný spôsob komunikácie. Samotné zasielanie správ prebieha prostredníctvom protokolu TCP. Spojenie medzi uzlami je však bezstavové a po každej výmene správ je ukončené. Komunikácia vždy prebieha formou výzva – odpoveď. Dokumentácia paketov správ je uvedená v prílohách.

Vrstva volania vzdialených procedúr – poskytuje možnosť zavolať vzdialenú procedúru iného uzla. Každé volanie procedúry zašle príslušnú správu vzdialenému uzlu a spracuje jeho odpoveď. Umožňuje taktiež využívať dátové typy v parametroch procedúry nezávisle na implementácii RPC.

Kontrolná vrstva – obsahuje implementáciu aplikačnej logiky, umožňuje vytvárať úlohy bežiace paralelne (pripojenie k sieti, vyhľadanie obsahu v sieti, publikovanie obsahu, získanie príspevku, atď.)

Vrstva užívateľského rozhrania – využíva kontrolnú vrstvu k ovládaniu lokálneho uzla a získavaniu informácií zo siete.

Pre implementáciu bol zvolený jazyk C++ s použitím frameworku Qt[2]. Toto riešenie zaisťuje svižný chod aplikácie a multiplatformovosť. Výsledkom implementácie je knižnica, nad ktorou je možné vybudovať textové alebo grafické užívateľské rozhranie.

Knižnica sa skladá z nasledujúcich modulov:

common.hpp – obsahuje definíciu kódov správ protokolu. Tento modul využívajú všetky moduly, ktoré pracujú priamo s dátami na úrovni zasielania správ.

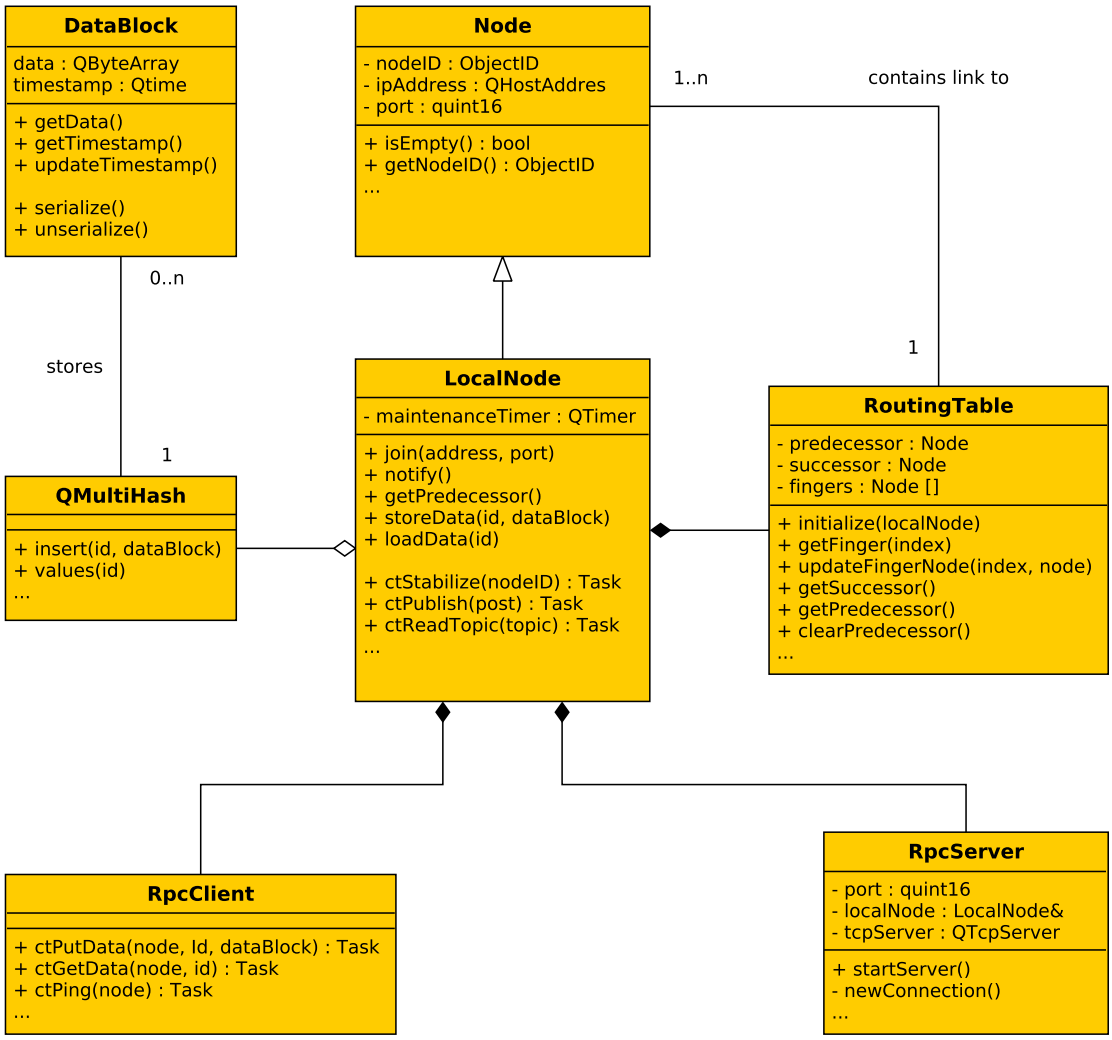
datablock.hpp – trieda popisujúca dátový blok, ktorý je možné uložiť do DHT. Typická veľkosť je 4 KiB. Dôležitou súčasťou je definícia serializácie objektu, ktorá umožňuje jednoduchý prenos po sieti. Každý dátový blok obsahuje taktiež časové razítko, ktoré môže byť použité k detekcii falšovania údajov o čase publikovania príspevku.

localnode.hpp – trieda reprezentujúca jeden uzol siete vytvorený na lokálnom počítači. Táto trieda obsahuje väčšinu logiky aplikácie. Je možné vytvoriť viacero lokálnych uzlov bežiacich na rôznom porte. Tento modul poskytuje všetky metódy k ovládaniu uzla a získavaniu dát zo siete.

node.hpp – trieda reprezentujúca uzol v sieti, neobsahuje žiadnu logiku, ale slúži ako identifikátor uzla (obsahuje okrem iného informácie ako IP adresa a port, na ktorom je uzol dostupný).

objectid.hpp – trieda reprezentuje identifikátor uzla alebo dátového bloku. Každý uzol je jednoznačne identifikovaný na základe výstupu hashovacej funkcie, ktorej vstup tvorí minimálne jeho IP adresa a port. Trieda nie je závislá na žiadnej konkrétnej hashovacej funkcii a je schopná použiť ľubovoľnú hashovaciu funkciu o ľubovoľnej dĺžke. Nad identifikátormi je možné vykonávať aritmetické operácie (Large Integer Arithmetic).

[2]Framework Qt (anglická výslovnosť cute) slúži pre tvorbu platformovo nezávislých aplikácií v natívnom kóde. Poskytuje abstraktné rozhranie pre tvorbu užívateľského rozhrania a prístup k systémovým zdrojom (ukladanie dát, sokety, atď.)

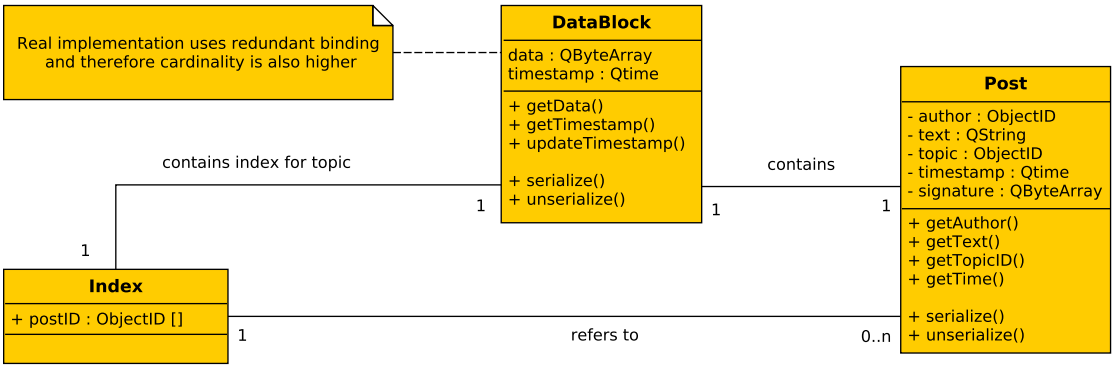


Obr. 4.3: Diagram tried popisujúci význam a vzťahy základných tried

post.hpp – reprezentácia jedného príspevku v sieti Liberty Share. Príspevok obsahuje minimálne identifikáciu autora, text príspevku, ID témy a čas publikovania zadaný autorom. Za účelom zabezpečenia vyššej bezpečnosti v sieti, musí byť každý príspevok podpísaný privátnym kľúčom autora príspevku, prípadne taktiež šifrovaný verejným kľúčom príjemcu alebo kľúčom derivovaným z názvu témy.

routingtable.hpp – implementácia smerovacej tabuľky. Tá je v pravidelných intervaloch obnovovaná a zachováva tak optimálne vlastnosti smerovania v sieti. Každý objekt RoutingTable je thread-safe čím zaručuje bezpečné použitie aj pri behu vo viacerých vláknach.

rpcclient.hpp – objekt RpcClient je možné použiť na volanie vzdialenej procedúry iného uzla. Následne sa objekt postará o odoslanie požiadavky na vykonanie procedúry a spracovanie odpovede. Každé volanie má nastavený maximálny časový limit na vykonanie procedúry.



Obr. 4.4: Diagram tried popisujúci princíp publikácie a evidencie obsahu

rpcserver.hpp – tento modul obsahuje implementáciu konkurentného serveru, ktorý spracuje RPC požiadavku, následne vykoná procedúru na lokálnom uzle a vráti odpoveď žiadateľovi.

taskmonitor.hpp – sleduje aktuálny počet pseudo-paralelných úloh bežiacich na lokálnom uzle. Tieto dáta môžu byť použité na distribúciu záťaže medzi vláknami, tvorbu štatistík, detekciu zahltenia alebo detekciu únikov pamäte (lokálne úlohy sú alokované na halde a tá musí byť po dokončení úlohy uvoľnená objektom, ktorý knižnicu využíva). Táto problematika je podrobne popísaná v sekcii 4.2.

Objektový návrh P2P siete podrobnejšie popisuje obr. 4.3. Ako prístupový bod k sieti Libery Share slúži trieda `LocalNode`. V diagrame si môžeme všimnúť metódy označené prefixom `ct` (Create Task). Tieto metódy slúžia pre vytváranie paralelne bežiacich úloh (väčšinou vyžadujú interakciu so sieťou a preto nemôže metóda vrátiť výsledok ihneď po zavolaní). Trieda `LocalNode` ukladá dátové bloky do dátovej štruktúry `QMultiHash`. Jedná sa o hashovaciu tabuľku umožňujúcu uložiť viacero hodnôt pod rovnakým kľúčom. Túto vlastnosť využijeme v prípade, kedy potrebujeme uložiť do indexu určitej témy viacero príspevkov.

Ďalej môžeme vidieť diagram popisujúci vzťahy medzi uloženými dátovými blokmi a spôsob, akým sú priradené príspevky k určitým témam (obr. 4.4). Publikovanie príspevku v sieti prebieha v nasledujúcich krokoch:

1. Autor požiada uzol, na ktorý ukazuje `ObjectID` príspevku o uloženie príspevku do DHT. V sieti je vytvorených viacero kópií príspevku, aby bolo zaručené jeho nájdenie aj v prípade výpadku viacerých uzlov v krátkom časovom okamžiku.
2. Autor požiada uzol, na ktorý ukazuje `ObjectID` témy o uloženie nového príspevku do indexu témy. Index obsahuje len jeho `ObjectID`, nie celý príspevok. Každá téma obsahuje viacero duplicitných indexov. Úroveň duplicity má pevne stanovenú počiatočnú hodnotu (napr. 128 kópií). Záťaž je možné pre viac frekventované témy vyvažovať pomocou použitia cache pamäte správ `GET_DATA` na smerujúcich uzloch.

4.2 Paralelizácia úloh

Uzol komunikujúci protokolom LSPP musí byť schopný konkurentne obsluhovať veľké množstvo klientov. Okrem toho, jeho rozhranie obsahuje úlohy, ktoré nie je možné vykonať okamžite (pripojenie k sieti, vyhľadanie obsahu, atď). Toto je častý problém sieťových aplikácií obecne. Pri zlom návrhu serveru je možné vyčerpať jeho systémové prostriedky už malým množstvom klientov. Tento problém sa nazýva C10K a jeho názov je odvodený od schopnosti obsluhovať desaťtisíc klientov súčasne [10]. Existujú dva hlavné prístupy ku konkurentnej obsluhe klientov:

- Spracovanie zmeny stavu siete asynchrónne pomocou definovania reakcie na určitú udalosť. Tento prístup sa nazýva *udalosťami riadená architektúra* (event-driven architecture).
- Spracovanie zmeny stavu siete synchrónne (blokujúce čakanie) a vytvorenie vlastného procesu alebo vlákna pre každého klienta. Je teda nutná *viacvláknová aplikácia* (multithreaded application).

Návrh pomocou použitia viacerých vlákien a synchrónneho čakania je často prehľadnejší a zachováva linearitu programu. Jeho hlavným nedostatkom je však nutnosť synchronizácie vlákien a vysoká spotreba pamäte (každý klient má vlastné vlákno a teda aj jeden zásobník o veľkosti napríklad 1 MiB). Na druhej strane, použitie asynchrónnej obsluhy udalostí nevyžaduje žiadne nadbytočné použitie pamäte. Aj z toho dôvodu je táto architektúra použitá vo výkonných webových serveroch ako nginx, JBoss Netty alebo Node.js.

Knižnica Liberty Share používa architektúru riadenú udalosťami, pre ktorú je vo frameworku Qt vstavaná podpora (koncept signálov a slotov). Vykonávanie pseudo-paralelných úloh tak prebieha prostredníctvom špeciálneho objektu nazývaného úloha (`Task`). Všetky úlohy sú vytvorené metódami s prefixom `ct` definované v objekte, ktorý za úlohu logicky zodpovedá. Používateľ knižnice tak môže spustiť ľubovoľný počet úloh (napríklad čítanie viacerých tém súčasne), pričom na dostupnosť výsledku bude upozornený prostredníctvom signálu `finished()`. V prípade chyby je namiesto signálu `finished()` emitovaný signál `error()` (viď kód 4.1).

Každá úloha musí byť dealokovaná objektom, ktorý ju vytvoril. Výnimku tvoria úlohy, pri ktorých tvorcu nezaujíma ich výsledok ani prípadný neúspech. Takéto úlohy môžu byť označené príznakom automatického odstránenia pomocou volania `setDeleteOnError(true)`. A pripojením signálu `finished()` k metóde Qt frameworku `deleteLater()`, ktorá úlohu odstráni:

```
Stabilize * stabilizeTask = ctStabilize();
connect(stabilizeTask, &Stabilize::finished,
        stabilizeTask, &Stabilize::deleteLater);
stabilizeTask->setDeleteOnError(true);
stabilizeTask->start();
```

Jednotlivé úlohy môžu vytvárať ďalšie úlohy a nastáva tak ich zreťazenie. Typické je vytvorenie lokálnej úlohy (napríklad úloha stabilizácie siete `Stabilize`), ktorú vytvára používateľ knižnice alebo lokálny uzol v pravidelných intervaloch. Tá následne vytvorí úlohu pre volanie vzdialenej procedúry `Notify`, pretože chce informovať iný uzol o zmene smerovacích informácií. Táto úloha odošle po sieti správu `NOTIFY`, ktorú spracuje úloha `RpcServerTask`

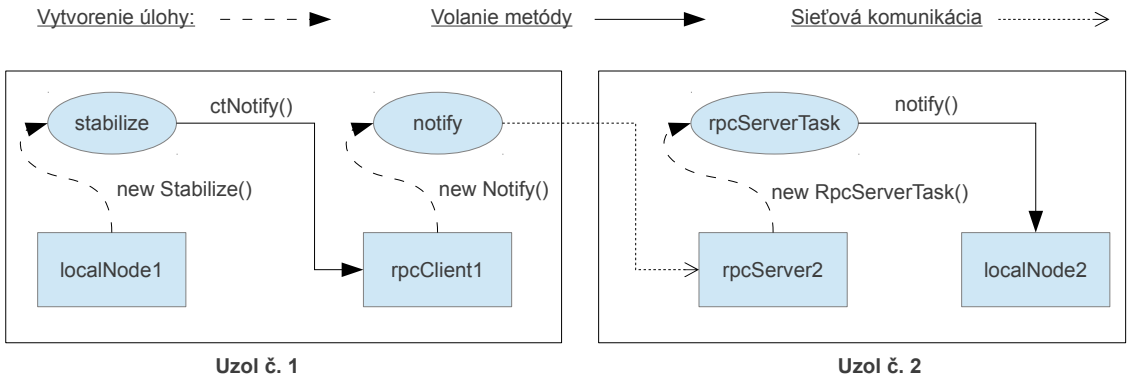
Kód 4.1 Práca s paralelnými úlohami

```
// get all posts recorded in index of the topic
while (topicIndex.hasNext())
    {
        ObjectID postID = i.next().getData();

        // get post by ID
        LocalNodeTask::GetData * getDataLt = NULL;
        getDataLt = localNode->ctGetData(postID);
        connect(getDataLtTmp, &LocalNodeTask::GetData::finished,
                this, &ReadTopic::getPostDone);
        connect(getDataLtTmp, &LocalNodeTask::GetData::error,
                this, &ReadTopic::handleLtError);

        getDataLt->start();
    }
```

vytvorená serverom. Posledná úloha vráti výsledok, ktorý sa následne deleguje cez predchádzajúce úlohy až k počiatočnej úlohe. To taktiež znamená, že úlohy zanikajú v opačnom poradí ako boli vytvorené (viď obr. 4.5).



Obr. 4.5: Koncept paralelného spracovania úloh a ich zreťazenie

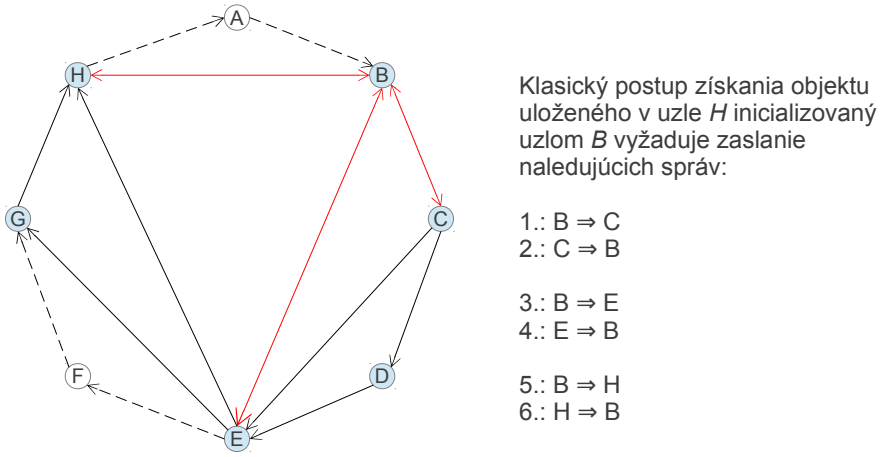
4.3 Zaistenie anonymity

Primárnou úlohou protokolu LSPP je umožniť jeho používateľom anonymné publikovanie obsahu v sieti. Všetky v súčasnosti používané techniky anonymizácie, uvedené v kapitole 2 sú aplikovateľné v P2P sieťach. Kľúčovým problémom bolo preto zvoliť techniku, ktorá poskytne dostatočnú mieru anonymity pri zachovaní čo najlepšej efektivity prenosu dát v sieti.

Ako vhodný kandidát sa javí technika Source-rewriting, pretože môže využiť vstavané možnosti smerovania vo vrstve prepojení Chord, ktorá je základom protokolu. Ostatné kategórie techník (Broadcast protocols a Dining-Cryptographers) by vyžadovali implementá-

ciu efektívneho broadcastu, prípadne vytvorenie iných bezpečných komunikačných kanálov. Klasické iteratívne získavanie obsahu v P2P sieťach používajúcich DHT prebieha v dvoch fázach (viď obr. 4.6, ktorý zobrazuje P2P sieť typu Chord a jej prepojenia) [18]:

1. Uzol B, ktorý požaduje získanie objektu v sieti kontaktuje[3] uzol C zo svojej smerovacej tabuľky a ten ho odkáže na ďalší uzol E, ktorý je v usporiadaní identifikátorov DHT bližšie k hľadanému cieľu. Uzol B iteratívne tento postup opakuje, až pokiaľ nenájde uzol H, ktorý má tento objekt uložený vo svojom lokálnom úložisku.
2. V poslednom kroku uzol B, ktorý chce objekt získať priamo kontaktuje uzol H a požiada ho o jeho zaslanie.



Obr. 4.6: Klasický iteratívny spôsob získania objektu

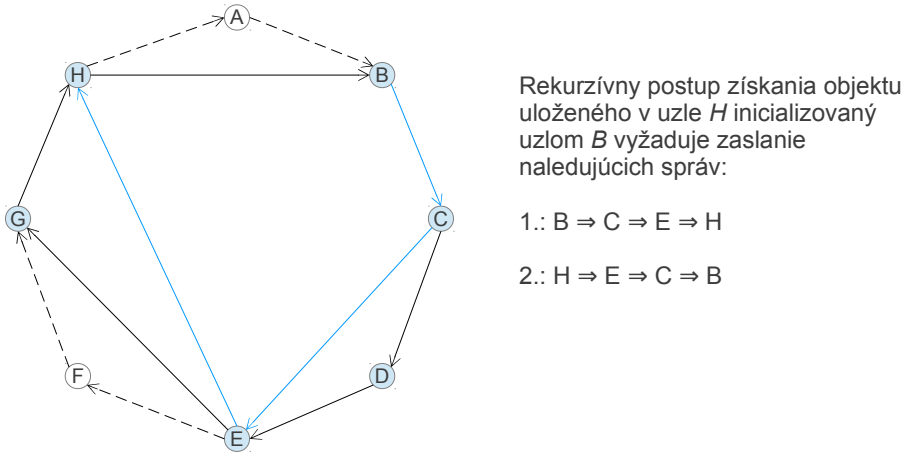
Hlavným nedostatkom tohoto prístupu je to, že uzol H vie kto je konzumentom obsahu asociovaným s daným objektom a zároveň uzol B vie, v ktorom uzle je daný objekt uložený.

Implementácia LSPP používa nasledujúcu rekurzívnu techniku získavania obsahu (viď obr. 4.7, ktorý zobrazuje rekurzívnu implementáciu získavania objektov):

1. Uzol B, ktorý požaduje získanie objektu v sieti, požiada o jeho získanie uzol C, ktorý má uložený vo svojich smerovacích tabuľkách. Táto žiadosť má rekurzívny charakter a ak uzol C nemôže tejto žiadosti vyhovieť, tak kontaktuje ďalší uzol E, ktorý je bližšie k hľadanému objektu v usporiadaní identifikátorov DHT. Toto smerovanie má zložitosť $O(\log N)$ a v ceste sa môžu vyskytnúť ktorékoľvek uzly $p \in V$.
2. Predposledný uzol E v tomto reťazci kontaktuje uzol H, ktorý obsahuje hľadaný objekt a požiada ho o jeho zaslanie. Následne prepošle tento objekt späť uzlu, od ktorého dostal žiadosť na získanie objektu. Takto sa hľadaný objekt dostane až k uzlu B.

Na obr. 4.7 môžeme vidieť, že uzly B a H nikdy nekomunikujú priamo. Táto rekurzívna implementácia nie je v P2P sieťach nijak výnimočná, avšak nikdy sa nepoužíva pre priamy prenos objektov, pretože zväčšuje potrebnú šírku prenosového pásma priamo úmernú dĺžke cesty a zároveň zanáša väčšiu latenciu do P2P siete. To sú však nedostatky, ktoré nám

[3] Notácia ⇒ znamená zaslanie jednej správy medzi dvoma uzlami.



Obr. 4.7: Rekurzívny spôsob získania objektu

v publikačných systémoch neprekážajú, pretože objem prenášaných dát je nižší a dáta samotné vyžadujú vysokú sieťovú redundanciu. Ak vybavíme uzly ležiace na ceste prenášaných dát dočasnou pamäťou cache, ktorá bude uchovávať prenášané objekty, získame prakticky 100 % využitie prenosového pásma (žiadny prenesený byte sa nezahodí a neprenesie zbytočne). V prípade použitia Onion routingu, techník využívajúcich broadcast alebo protokolu DC-Net by sa určitá časť dát vždy zahodila a dostupné prenosové pásmo by sme nevyužili efektívne.

Veta 1. *V každej úplnej (stabilizovanej) sieti $G = (V, E)$ typu Chord používajúcej rekurzívne získavanie objektov, existuje cesta P medzi ľubovoľnými uzlami $x < y < z$ ak existuje hrana $e = (y, z)$.*

Dôkaz. Cestu môžeme zostrojiť 2 spôsobmi:

1. Nech uzol x je počiatok cesty a uzol z je jej koniec, potom cestu P zostrojíme ako $P = (v_0, e_0, v_1, e_1, \ldots e_n, v_n)$, kde $v_0 = x$, $e_0 = (x, \text{successor}(x))$, $v_i = \text{successor}(v_{i-1}), e_i = (v_i, v_{i+1})$, $e_n = (y, z), v_n = z$. Táto cesta vedie po pomyselnom kruhovom výseku medzi x a z.
2. Nech uzol x je počiatok cesty a uzol z je jej koniec, potom cestu P zostrojíme ako $P = (v_0, e_0, v_1, e_1, \ldots e_n, v_n)$, kde $v_0 = x$, $e_0 = (x, \text{finger}(x))$, $v_i = \text{finger}(v_{i-1}), e_i = (v_i, v_{i+1})$, $e_n = (y, z), v_n = z$. Táto cesta vedie cez uzly v smerovacích tabuľkách uzlov (fingers).

□

Veta 1 implikuje, že ak obdrží uzol z správu od uzla y, jej originálnym autorom x môže byť ktorýkoľvek uzol v sieti, ktorý je predchodcom y. Ak sú zároveň všetky cesty zo všetkých uzlov rovnako pravdepodobné, tak smerovanie zabezpečuje taktiež anonymitu.

Ak útočník pozná úplnú štruktúru siete a disponuje dostatočným množstvom zachytených správ, môže sa pokúsiť cestu zrekonštruovať. Tomuto môžeme predchádzať zavedením určitej miery nedeterminizmu do smerovacích algoritmov a zabezpečiť tak čiastočnú anonymitu.

Na to, či je toto smerovanie dostatočne anonymné pre použitie v protokole LSPP sa pozrieme v kapitole 5.

4.4 Zaistenie necenzúrovateľnosti

Druhá dôležitá vlastnosť, ktorú protokol LSPP zaisťuje je necenzúrovateľnosť obsahu publikovaného v sieti. Klasickým riešením je použitie distribuovanej architektúry spolu s vysokou redundanciou dát. Ideálne je taktiež, ak sú dáta distribuované rovnomerne na viacerých uzloch v rôznych štátoch. K tomuto účelu sú použité dva typy redundancie v sieti Libery Share:

- *Statická redundancia* je určená a spravovaná uzlom pri publikovaní obsahu (uzol si zvolí koľko kópií vytvorí pri prvom publikovaní). Tento typ redundancie je určený konštantou definovanou v klientskom software.
- *Dynamická redundancia*, ktorá vzniká pri prenose dátových blokov jednotlivými uzlami a zvyšuje sa každým prenosom bloku. Má význam predovšetkým pri starších príspevkoch, ktoré sa už nemenia a je možné ich uložiť do cache pamäte uzla (na rozdiel od indexu, ktorý sa mení často). Bloky, o ktoré je väčší záujem majú aj väčšiu redundanciu.

Statická redundancia využíva rovnomernú distribúciu objektov v DHT pomocou hashovacej funkcie. Knižnica Liberty Share používa generické rozhranie pre prístup k identifikátorom a preto je možné k tomuto účelu použiť ľubovoľnú hashovaciu funkciu. Súčasná implementácia používa štandard SHA-2 nasledujúcim spôsobom:

$$dataBlockID_i = \text{SHA-256}(dataBlock \mid i), \quad i \in \{0, 1, 2 \dots R-1\} \tag{4.1}$$

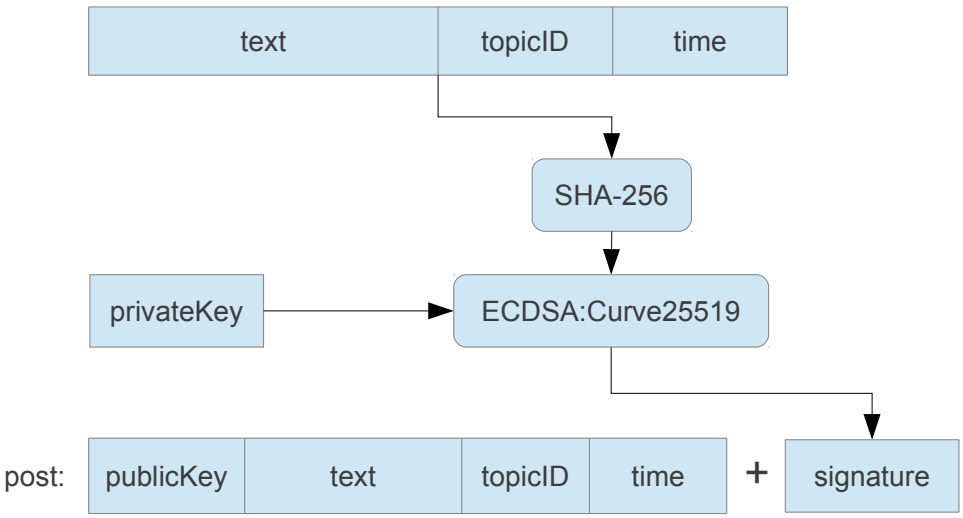
kde $dataBlockID_i$ je identifikátor pre prístup k i-tej inštancii dátového bloku, *dataBlock* je postupnosť bytov dátového bloku, symbol $\mid$ značí konkatenáciu a R je miera redundancie (počet kópií). Existujú aj pokročilejšie metódy zaistenia redundancie, ktoré môžu byť implementované dodatočne [9].

4.5 Bezpečnosť

Bezpečnosť patrí medzi najdôležitejšie aspekty sieťových protokolov. Obzvlášť v prostredí necentralizovaných P2P sietí patrí bezpečnosť a odolnosť voči rôznym útokom k aktívnym oblastiam výskumu. Medzi kľúčové bezpečnostné funkcie protokolu LSPP patria okrem anonymity a necenzúrovateľnosti aj nasledujúce:

- Zaistiť jednoznačnú identifikáciu príspevkov od rovnakého autora, ktorého identita musí byť stále utajená.
- Umožniť vytvárať súkromné témy, ku ktorým nemajú prístup neoprávnení čitatelia.
- Umožniť publikovať súkromný príspevok, ktorý môže prečítať iba konkrétny adresát.
- Zabrániť falšovaniu času publikovania príspevku.

K tomuto účelu použijeme asymetrické a symetrické šifrovacie algoritmy. Predovšetkým asymetrické algoritmy sú vhodné k anonymnej identifikácii autora príspevku pomocou podpisu jeho privátnym kľúčom, pričom verejný kľúč slúži ako samotný identifikátor.



Obr. 4.8: Podpisovanie príspevku pred publikovaním v sieti Libery Share

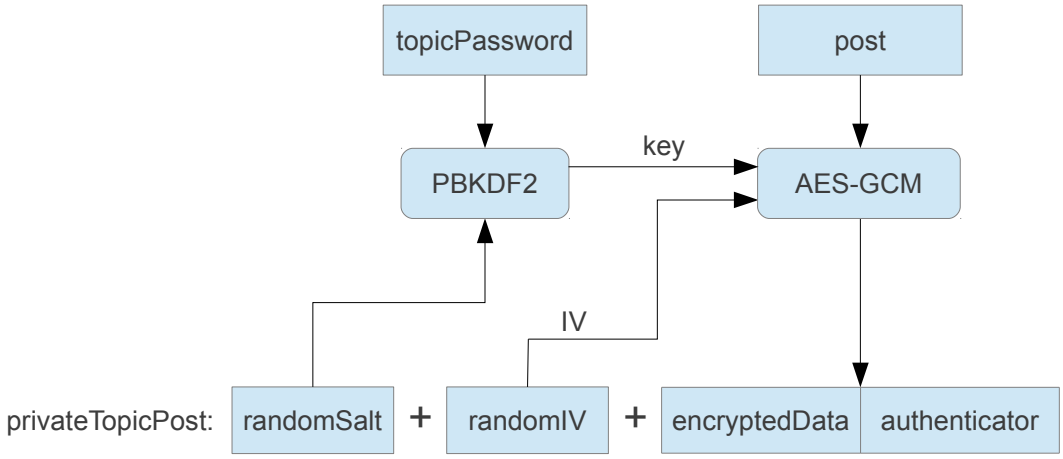
Na obr. 4.8 môžeme vidieť schému znázorňujúcu proces podpisovania príspevku pred jeho publikovaním. Vo fáze návrhu bola zvolená asymetrická kryptografia nad eliptickými krivkami reprezentovaná algoritmami ECDSA a ECIES. Tieto algoritmy poskytujú dobrú mieru bezpečnosti, krátku dĺžku kľúčov a dostatočnú rýchlosť. Alternatívne je možné použiť algoritmus RSA a mnoho ďalších, ktoré disponujú taktiež vhodnými vlastnosťami. Podpisovanie príspevkov zabezpečuje integritu, autentifikáciu a nepopierateľnosť [13].

Keďže navrhnutá P2P sieť neobsahuje žiadne centrálne servery a nedisponujeme ani možnosťou vytvorenia dôveryhodnej certifikačnej autority, nemôžeme pri podpise použiť certifikát autora príspevku. Namiesto toho je povinný každý klient udržovať zoznam verejných kľúčov autorov príspevkov, ktorí už predtým v sieti publikovali. Na základe tohoto zoznamu je schopný bezpečne identifikovať autora príspevku. Aby bol tento proces pre užívateľa softwaru jednoduchší, mal by mať možnosť transparentne mapovať verejné kľúče na textové pseudonymy zvolené používateľom (i.e., malo by byť možné verejný kľúč pomenovať).

Na obr. 4.9 môžeme vidieť schému šifrovania príspevku pred publikovaním v súkromnej téme. Súkromná téma je chránená heslom a príspevky v tejto téme môžu čítať iba užívatelia, ktorí heslo poznajú. Každá téma v sieti Liberty Share je jednoznačne identifikovaná pomocou textového identifikátoru *topicName* v nasledujúcom formáte:

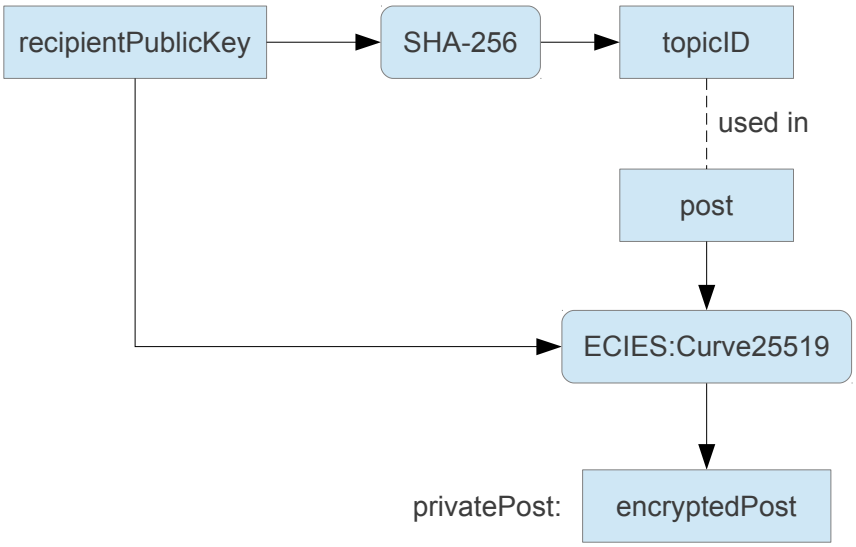
`+<názov>:<heslo>`

Platnými identifikátormi sú teda napríklad `+ForVipOnly:H4kp7moVm1` pre súkromnú tému alebo `+SportsCars` pre verejnú tému. Identifikátor *topicID* reprezentuje výsledok hashovacej funkcie aplikovanej na textovú reprezentáciu: $topicID = \text{SHA-256}(topicName)$. Zvýšenú pozornosť treba venovať hodnotám *randomSalt* a *randomIV*, ktoré sa nesmú opakovať a mali by byť preto vytvorené kryptograficky bezpečným generátorom náhodných čísel.



Obr. 4.9: Vytvorenie príspevku v súkromnej téme

Nemenej dôležitou funkciou je publikovanie príspevku, ktorý vidí iba jeden konkrétny používateľ. Túto funkciu je možné chápať aj ako zaslanie súkromnej správy. Postup vytvorenia príspevku, ktorý môže čítať iba uzol vlastniaci príslušný privátny kľúč môžeme vidieť na obr. 4.10.



Obr. 4.10: Vytvorenie súkromného príspevku

Všetky doteraz uvedené bezpečnostné funkcie sú založené na overených algoritmoch a preto ich môžeme považovať za dostatočne bezpečné. Pre overenie času publikovania príspevku však nemáme k dispozícii nástroj alebo techniku, ktorá by nevyžadovala použitie spoľahlivej tretej strany TSA (Time Stamping Authority). Ako jediný spoľahlivý subjekt môžeme použiť uzly, ktoré majú vo svojom lokálnom úložisku uložený okrem samotného príspevku aj čas jeho publikovania. K tomuto účelu môžeme použiť medián získaných časov:

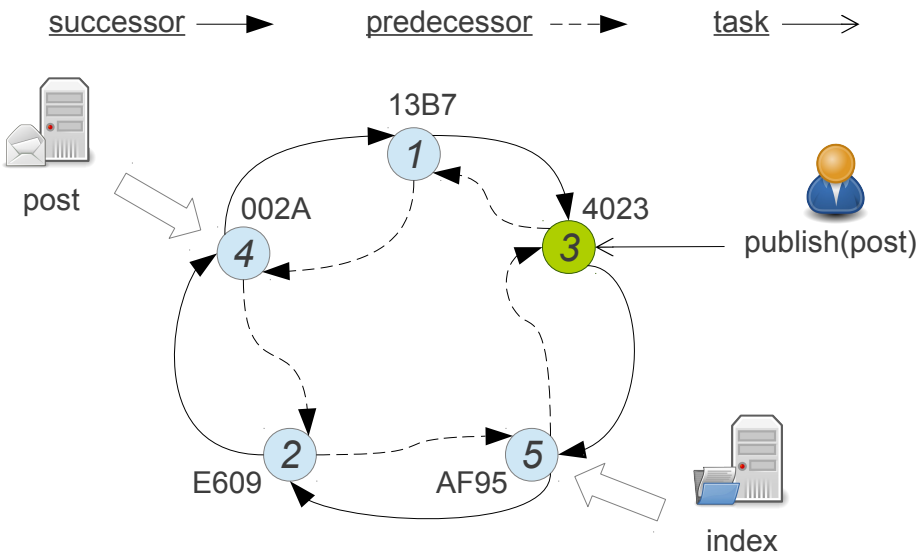
$$time = \text{median}(T_{blok} \cup \{t_{author}\}) \tag{4.2}$$

Kde T_{blok} je množina časov vytvorená z časových razítok v získaných redundantných blokoch a t_{author} je čas uvedený a podpísaný autorom príspevku.

4.6 Použitie

Knižnica Liberty Share, ktorú je možné nájsť v prílohách je použiteľná jednak ako plnohodnotná implementácia LSPP protokolu (bez kryptografických funkcií), ale taktiež ako nástroj k vytváraniu virtuálnych P2P sietí na lokálnom počítači. Druhý spôsob je vhodný predovšetkým pre testovacie a simulačné účely. V oboch prípadoch pristupujeme k P2P sieti prostredníctvom lokálneho uzla, ktorého chovanie môžeme ovplyvňovať pomocou vstavaného API. Prvý krok je pripojenie k sieti prostredníctvom bootstrap uzla a následne môžeme publikovať príspevok pomocou lokálnej úlohy `Publish` (viď príklad 4.2).

Pomocou úlohy `ReadTopic("+MyFirstTopic")` môžeme získať všetky príspevky v danej téme z ktoréhokoľvek uzla. Pre testovacie účely môžeme využiť taktiež volanie `getReport()` na ktoromkoľvek uzle, ktoré vytvorí textovú reprezentáciu stavu uzla typu `QString`. Tá obsahuje informácie o smerovacích tabuľkách uzla.



Obr. 4.11: Uloženie príspevku v sieti Liberty Share bez redundancie

Na obr. 4.11 vidíme stav siete po publikovaní nášho ukážkového príspevku. Pre zachovanie prehľadnosti je príspevok publikovaný bez redundancie. Pre úplnosť sú kompletné smerovacie informácie uvedené v tabuľke 4.1. Hodnoty F_x reprezentujú hodnoty typu finger tak, ako sú definované vo vrstve prepojení Chord. Pre zlepšenie prehľadnosti boli pôvodné identifikátory skrátené z 256 bitov na hexadecimálny prefix o dĺžke 16 bitov.

Port	NodeID	Predecessor	Successor	F_0	F_1	F_2
9001	13b7	002a	4023	af95	af95	4023
9002	e609	af95	002a	af95	4023	13b7
9003	4023	13b7	af95	e609	af95	af95
9004	002a	e609	13b7	af95	af95	4023
9005	af95	4023	e609	4023	002a	e609

Tabuľka 4.1: Smerovacie tabuľky

Kód 4.2 Ukážka použitia knižnice Liberty Share

```
TestClass::TestClass() :
  // vytvor lokálne uzly na portoch 9001-9005
  node1(9001), node2(9002), node3(9003), node4(9004), node5(9005)
{
  // obsluha signálu joinDone() pre uzol 3
  connect(node3, &LocalNode::joinDone, this, &TestClass::joinDone);
  connect(node3, &LocalNode::joinFailed, this, &TestClass::joinFailed);
  ...

  // pripoj ostatné uzly k uzlu 1
  // uzol 1 je tzv. bootstrap node
  node2.join(QHostAddress::LocalHost, node1.getPort());
  node3.join(QHostAddress::LocalHost, node1.getPort());
  ...
}

// s uzlom môžeme pracovať až po úspešnom pripojení k sieti
void TestClass::joinDone()
{
  qDebug() << "node3 is connected";

  Post post("John Doe", "+MyFirstTopic", "A text of my new post.");

  // vytvorime lokálnu úlohu Publish
  publishLt = node2.ctPublish(post);
  connect(publishLt, &LocalNodeTask::Publish::finished,
    this, &TestClass::publishDone);
  connect(publishLt, &LocalNodeTask::Publish::error,
    this, &TestClass::publishFailed);
  publishLt->start();
}

void TestClass::publishDone()
{
  qDebug() << "The post has been successfully published";
  publishLt->deleteLater();
}
```

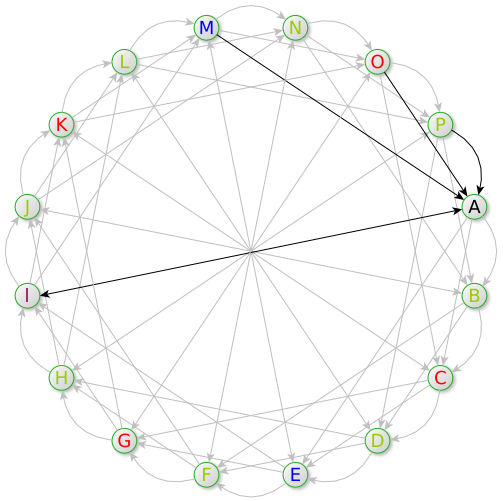
Kapitola 5

Analýza protokolu LSPP

Rovnako ako v prípade mnohých iných protokolov nie je publikačný protokol LSPP univerzálnym riešením všetkých problémov anonymných P2P sietí. V tejto kapitole sa pokúsime identifikovať jeho slabiny a silné stránky, čím jasne definujeme oblasti, v ktorých je jeho aplikácia vhodná, prípadne nevhodná. Pretože smerovanie v protokole LSPP je založené na vrstve prepojení Chord, budú naše zistenia aplikovateľné aj na tento protokol.

5.1 Anonymita

Prvá a zároveň kľúčová vlastnosť, na ktorú sa pozrieme je zaistenie anonymity. Už v definícii požiadaviek na protokol sme určili, že publikačný systém by mal zabezpečovať úplnú anonymitu. V odôvodnených prípadoch by sme mohli akceptovať aj čiastočnú anonymitu. V sekcii 4.3 sme ukázali, že autorom ľubovoľnej správy protokolu LSPP môže byť ktorýkoľvek uzol, pričom tento uzol nekomunikuje s príjemcom správy priamo. Zaručuje však tento systém úplnú alebo čiastočnú anonymitu? A voči ktorým útokom je, respektíve nie je odolný?



Obr. 5.1: Určenie podmnožín autorov správ

5.1.1 Deterministické smerovanie

Predpokladajme plne obsadenú stabilizovanú vrstvu prepojení typu Chord, kde $pid_A = 0$, $pid_B = 1$, $\ldots pid_P = 15$. Ďalej predpokladajme, že uzol A, príjme správu m (správy budeme považovať za šifrované verejným kľúčom A). Na základe algoritmu smerovania môžeme s istotou povedať, že posledný uzol, ktorý správu doručí uzlu A môže byť len uzol $y \in \{I, M, O, P\}$ (viď obr. 5.1).

Ak vypočítame najoptimálnejšiu trasu smerovania z každého uzla x a vytvoríme dvojice (y, x), kde x je počiatočný uzol a y je uzol, z ktorého obdržal správu uzol A, tak získame množinu dvojíc Ω:

$$\begin{array}{rcl} (P,\alpha) & | & \alpha \in \{P, N, L, J, H, F, D, B, \} \\ (O,\beta) & | & \beta \in \{O, K, G, C\} \\ (M,\gamma) & | & \gamma \in \{M, E\} \\ (I,\delta) & | & \delta \in \{I\ \} \end{array}$$

Pravdepodobnosti prijatia určitej správy od uzla x sa teda líši v závislosti na tom, do ktorej množiny uzol x patrí a pre pravdepodobnosť p_x, že správa pochádza práve z uzla x teda platí:

$$\begin{aligned} p_\alpha &= \frac{1}{8} = 0,125 \\ p_\beta &= \frac{1}{4} = 0,25 \\ p_\gamma &= \frac{1}{2} = 0,5 \\ p_\delta &= \frac{1}{1} = 1 \end{aligned}$$

Jasne teda vidíme, že najväčšej miere anonymity sa môžu tešiť uzly zastúpené symbolom α patriace do prvej množiny a naopak v prípade uzla I nie je možné hovoriť o anonymite vôbec. Tieto pravdepodobnosti sú platné, ak je príjemcom správy uzol A. Ak bude príjemcom správy iný uzol, tak sa príslušnosť autorov správ do jednotlivých množín zmení. Tento stav je teda jasne nedostačujúci, pretože každý uzol sa môže ocitnúť v pozícii, kedy bude jeho miera anonymity čiastočná alebo dokonca žiadna.

5.1.2 Nedeterministické smerovanie

V reálnych aplikáciach nie je nikdy sieť plne obsadená a vo väčších sieťach s vysokou frekvenciou pripájania a odpájania uzlov zo siete nie je taktiež pravdepodobné, že by smerovanie prebiehalo vždy optimálne (neoptimálne smerovanie zmení príslušnosť do množín). Napriek

tomu nemôžeme tento problém ignorovať, pretože otvára dvere množstvu štatistických útokov.

Potrebujeme sa teda zbaviť silne deterministického chovania smerovacích algoritmoch a nahradiť ho nedeterministickým algoritmom s rovnakou zložitosťou. Kľúčovou vlastnosťou väčšiny anonymizačných techník, s ktorými sme sa stretli v kapitole 2 je náhodnosť ich chovania. Nahradíme teda položky smerovacej tabuľky nasledujúcim spôsobom, kde funkcia *rand* označuje zaokrúhlenie na najbližšie celé číslo, pričom $0,5$ zaokrúhľujeme nahor:

$$finger_i = round(pid + 2^{i-1} + rand) \bmod 2^m \quad | \quad rand \in \langle -(2^{i-2}), 2^{i-2}), \ rand \in \mathbb{R} \tag{5.1}$$

Pri náhodnej voľbe $\max(rand)$ alebo $\min(rand)$ sa teda môže stať, že budeme potrebovať namiesto dvoch skokov jeden alebo namiesto jedného dva. Zložitosť nám tým pádom v najhoršom prípade klesne z pôvodnej zložitosti smerovania $O(\log_2 N)$ na $O(2 \log_2 N)$ a v najlepšom na $O(0,5 \log_2 N)$. Pravdepodobnosť týchto krajných prípadov p_w je však veľmi malá a pre dĺžku identifikátora v bitoch $m = 4$ a počet uzlov $N = 16$:

$$p_w = \prod_{x=0}^{m-1} \frac{1}{2^x} = \prod_{x=0}^{3} \frac{1}{2^x} = \frac{1}{64} \doteq 0,0156 \tag{5.2}$$

Oba prípady sú teda nepravdepodobné a vďaka rovnomernej distribúcii identifikátorov zostane priemerná zložitosť zachovaná. Z obr. 5.1 je zrejmé, že táto zmena spôsobí, že správa z každého uzla x môže byť doručená do uzla z aj cez cesty iné ako optimálne. Zmení sa taktiež množina Ω a pravdepodobnosti p_x, ale ešte stále nenastane ich vyrovnanie.

5.1.3 Simulácia

Aby sme zistili anonymizačné vlastnosti nedeterministického smerovania, vytvoríme jeho simulátor (viď kód B.1) a pre zistenie pravdepodobností použijeme metódu Monte Carlo[1] (empirické riešenie je v tomto prípade výrazne jednoduchšie ako analytické). Výsledkom simulácie je tabuľka udávajúca početnosť správ zaslaných z uzla x, pričom posledný uzol, ktorý doručí správu uzlu z je uzol y.

V tabuľke B.1 vidíme výsledky prvej simulácie s použitím vzorca 5.1. Zaujímavý je predovšetkým riadok, kedy je správa adresovaná uzlu z doručená uzlom $y = F$. Týmto uzlom môžu byť doručené len správy z uzla B alebo F. Tento problém je spôsobený použitím príliš dlhých skokov, kedy sa nestihne prejaviť rekurzia smerovania a správa je doručená priamo zdrojovým uzlom. V tabuľke B.2 vidíme výsledky simulácie, kedy je maximálna dĺžka skoku skrátená na polovicu, čo odpovedá zložitosti $O(\log_2 N + 2)$. Početnosť a teda aj pravdepodobnosti p_x sú viacej vyrovnané, ale stále nie sú rovnaké. Takýto systém poskytuje len čiastočnú anonymitu a úplnú anonymitu by sme dosiahli až po znížení kroku na vzdialenosť 1, kedy by pre pravdepodobnosti platilo:

$$p_A = p_B = p_C \ldots p_P = \frac{1}{N} \tag{5.3}$$

[1] Trieda algoritmov používaných pri simulácii. Jedná sa o stochastickú metódu využívajúcu pseudonáhodné čísla k zisteniu určitých štatistických vlastností systému.

ale zložitosť smerovania by sa tak zhoršila na $O(N)$.

Ak autor posiela správy, ktoré obsahujú údaj o pseudoanonymnej identite uzla (napr. verejný kľúč), tak môže útočník po zozbieraní dostatočného množstva správ využiť štatistické údaje o uzloch y, ktoré mu správu doručili a na základe korelácie medzi predpokladaným počtom doručených správ z jednotlivých uzlov y a skutočným počtom môže odhadnúť skutočnú identitu autora správ. Presnosť tohoto útoku sa zvyšuje s počtom zachytených správ a frekvenciou zmeny smerovacích tabuliek.

5.1.4 Rozšírenie protokolu LSPP

Za účelom zníženia účinnosti štatistických útokov by sme mohli ďalej modifikovať smerovací algoritmus a získať tak lepšie rozloženie pravdepodobností. Výsledkom tejto snahy by však bolo v každom prípade zhoršenie vlastností smerovania a zväčšenie latencie.

Iné riešenie tohoto problému je inicializovať odoslanie správy z náhodného miesta vo vrstve prepojení Chord. Tým pádom, nebude existovať žiadna štatistická závislosť medzi uzlami x a y, čo nám podľa očakávaní potvrdil aj výsledok simulácie v tabuľke B.3. Takto prenášané správy však musia byť šifrované (aby sme predišli útokom zo strany uzlov ležiacich na prenosovej ceste), čo je v rozpore s naším pôvodným predpokladom o použití cache pamäte dátových blokov pri ich prenose v sieti.

Jedno z možných riešení je preto nešifrovať prenášané správy, ale pridať do protokolu novú anonymizačnú vrstvu založenú na algoritme Onion Routing, ktorá bude fungovať nasledujúcim spôsobom:

1. Pomocou broadcastu na aplikačnej úrovni bude každý uzol oznamovať jeho prítomnosť v sieti spolu s jeho IP adresou a portom, na ktorom beží. Broadcast môžeme implementovať buď priamo pomocou broadcastu na existujúcej vrstve prepojení [8] alebo pridaním nových neštrukturovaných prepojení.
2. Uzol, ktorý chce poslať po sieti správu obsahujúcu jeho pseudoanonymnú identitu (verejný kľúč) odošle správu prostredníctvom protokolu Onion Routing používajúceho k náhodne zvolených uzlov z broadcastu. Hodnota k môže byť zvolená na základe dôležitosti ochrany správy. Príjemcom tejto správy je náhodný uzol v sieti, ktorý bude túto prijatú správu ďalej smerovať do cieľového uzla pomocou smerovacieho algoritmu v protokole LSPP.

Onion Routing je veľmi optimálny protokol, ktorý poskytuje úplnú anonymitu a pre pravdepodobnosť p_x, bude platiť v prípade, že broadcast obsahuje všetky uzly:

$$p_x = N^{-1} \tag{5.4}$$

Ďalším rozšírením protokolu môže byť dynamické monitorovanie veľkosti siete. To v prípade neštrukturovaných sietí nie je jednoduché, ale v prípade sietí štrukturovaných môžeme využiť rozloženie identifikátorov *pid* v priestore identifikátorov. Túto informáciu môžeme použiť k určovaniu vhodnej redundancie alebo k prispôsobeniu anonymizačných mechanizmov (v menších sieťach môžeme použiť techniky, ktoré vo väčších nie sú efektívne – e.g., broadcast). Odhad zaplnenia siete N_g teda vypočítame ako:

$$N_g = \frac{2^m}{2\ |rand_{sid} - sid_{successor}|} \quad rand \in \langle 0, 2^m - 1 \rangle, rand \in \mathbb{Z} \tag{5.5}$$

kde $rand_{sid}$ je náhodný identifikátor objektu a $sid_{successor}$ je uzol nájdený pomocou volania úlohy `FindSuccessor`, ktorý za tento identifikátor zodpovedá. Táto metóda nie je veľmi presná a preto je potrebné vykonať viacero pokusov, z ktorých vypočítame aritmetický priemer. Namiesto úloh `FindSuccessor` môžeme použiť už existujúce hodnoty identifikátorov zo smerovacích tabuliek a znížiť tak záťaž siete.

5.1.5 Cache

Správy prenášané protokolom LSPP nepoužívajú šifrovanie prenášaných správ (to sa netýka šifrovania príspevkov). Dôvodom k tomuto kroku je šetrenie prenosového pásma, pričom chceme čo najviac využiť prenášané dáta. Výsledky vyhľadávania príspevkov v sieti tak môžu byť uložené do lokálnej cache pamäti uzla a po určitú dobu uchovávané. Tento krok zvyšuje rýchlosť vyhľadávania a zvyšuje redundanciu dát v sieti.

Každý príspevok v cache pamäti poskytuje potenciálnemu útočníkovi viac informácií o ceste, po ktorej boli dáta prenášané. V prípade prenosu správ tak, ako sme to popísali v sekcii 5.1.4, toto nepredstavuje výrazné bezpečnostné riziko. Útočník by aj po úspešnej rekonštrukcii cesty našiel maximálne výstupný bod Onion Routingu.

Napriek tomu je vhodné nesprístupňovať objekty v cache pamäti okamžite, ale až po viacnásobnom prechode objektu uzlom. Tento princíp je známy z techniky Mix-net popísanej v sekcii 2.1 s jej použitím sa rekonštrukcia cesty stane obtiažnejšou. Predpokladajme sprístupnenie objektu po n prechodoch, potom po každom sprístupnení objektu vedie cez uzol n ciest, čo je pre priemernú dĺžku cesty k rovné počtu možných ciest V':

$$V'_k(n) = n^k = n^{\log_2(N)} \tag{5.6}$$

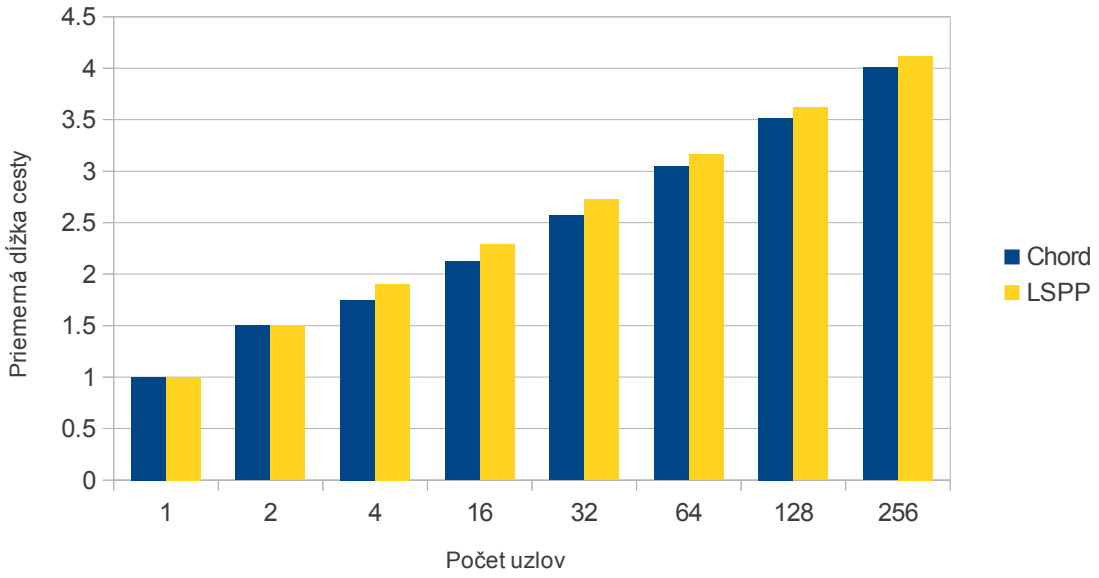
Už pre $n = 2$ je $V'_k(2) = 2^{\log_2(N)} = N$ a teda ciest je toľko čo uzlov. Sprístupnenie po 2. prechode sa teda javí ako dostatočné.

5.2 Efektivita

Efektivita protokolu LSPP vychádza z vrstvy prepojení Chord. Jedinou výraznejšou zmenou smerovacieho algoritmu, ktorá ovplyvňuje efektivitu je použitie nedeterministického smerovania. Ako sme už ukázali v sekcii 5.1.2, táto zmena má len minimálny dopad na efektivitu. O tejto skutočnosti sa môžeme presvedčiť na obr. 5.2.

5.3 DoS útoky a spam

Podobne ak väčšina iných P2P sietí je protokol LSPP, tak ako sme ho popísali v predchádzajúcich sekciách náchylný k množstvu DoS útokov. Pretože užívatelia používajú pseudoanonymné identifikátory, ktorých si môžu vytvoriť neobmedzený počet, nie je možné limitovať



Obr. 5.2: Priemerná dĺžka cesty počas simulácie

systémové prostriedky na základe identity. Za predpokladu, že má každý užívateľ pridelenú unikátnu IP adresu, môžeme v bežnej P2P sieti určité limity prideliť samotným adresám. V prípade anonymných sieti však IP adresa užívateľa nie je známa a preto tento spôsob nie je možné použiť. Potenciálny útočník teda môže vykonať nasledujúce útoky:

- Vyčerpá dostupné miesto v lokálnom úložisku určitého uzla pomocou ukladania nezmyselných dát a prinúti ho tak zahadzovať už uložené správy.
- Zahltí určitý uzol nezmyselnými správami a zabráni mu tak v spracovaní ostatných správ.
- Publikuje nezmyselné príspevky v jednej alebo viacerých témach a tá sa tak stane pre čitateľa témy nezrozumiteľnou a neprehľadnou.

Prirodzene by sme mohli nájsť viacero útokov tohoto typu. K riešeniu tohoto problému máme k dispozícii minimálne tri techniky, pričom cieľom každej je prinútiť autora správy vynaložiť určité úsilie alebo prostriedky pred odoslaním správy:

- Autor správy musí dokázateľne vykonať určité množstvo výpočtov pred tým, ako mu bude povolené publikovanie príspevku. Jedná sa teda o techniku Proof-of-work, ktorej príkladom je napr. Hashcash [2] použitý v systéme Bitcoin.
- Autor správy musí za publikovanie správy zaplatiť určitý obnos finančných prostriedkov, napríklad vo forme mikrotransakcie.
- Autor správy musí vyriešiť strojovo neriešiteľnú úlohu. Príkladom takejto úlohy je CAPCHA. Táto úloha môže byť vytvorená strojom, ale vyriešiť ju dokáže iba človek. V súčastnosti sa rozpoznávacie algoritmy neustále vylepšujú a tak môže byť vytvorenie tejto úlohy čoraz náročnejšie.

5.4 Decentralizované vytvorenie identity

Útoky na *nodeID* (*pid*) môžeme riešiť v protokole LSPP podobne ako DoS útoky. Rozdiel spočíva vo frekvencii jednotlivých úloh, pričom identifikátor uzla (identitu v sieti) si uzol vytvára iba raz, ale čítanie a publikovanie správ prebieha neustále. Dôsledkom tohoto rozdielu je to, že pri vytvorení identity môžeme použiť viac techník ochrany, ale pri publikovaní príspevku by mala byť záťaž na užívateľa nižšia. Identita uzla (*pid*) nesúvisí s pseudoidentitou používateľa (verejný kľúč). Vytvorenie identity uzla v sieti Liberty Share by sme mohli implementovať nasledujúcim spôsobom:

1. Žiadateľ o identitu v sieti vypočíta identifikátory správcov, ktorí budú zodpovední za vytvorenie a správu novej identity. Počet správcov I je konštantný. Identifikátory vypočítame rekurzívne nasledujúcim spôsobom (h značí ľubovoľný dostatočne bezpečný hashovací algoritmus):

$$\begin{aligned} pid_0 &= h(ipAdresa) \\ pid_i &= h(pid_{i-1}) \end{aligned}$$

2. Žiadateľ kontaktuje všetkých I správcov a požiada o vytvorenie novej identity (pid, $ipAdresa$, k_{pb}), kde pid je budúci identifikátor žiadateľa a k_{pb} je verejný kľúč žiadateľa. Každý z oslovených správcov overí IP adresu a požiada žiadateľa o vyriešenie úlohy umožňujúcej Proof-of-work.

3. Po overení výsledku úlohy vytvorí každý zo zoznamu správcov s identifikátormi (pid_0, pid_1, ..., pid_{I-1}) sieťové spojenie s ostatnými správcami v zozname.

4. Každý správca požiada žiadateľa o vyriešenie jedného písmena testu CAPTCHA. Tieto písmená budú na strane žiadateľa zobrazené vedľa seba vo forme sériovo spojených obrázkov.

5. Každý správca odošle výsledok testu broadcastom ostatným správcom. V prípade, že je počet správne vyriešených písmen väčší ako minimálna povolená hodnota, bude výsledok akceptovaný. Inak nastane návrat na krok 2.

6. Každý správca odošle broadcastom náhodnú hodnotu n_i, šifrovanú náhodným symetrickým kľúčom k_i (šifra musí byť autentifikovaná).

7. Po prijatí poslednej hodnoty $E_{k_i}(n_i)$ odošlú správcovia použité kľúče k_i broadcastom ostatným správcom. Nechceme totiž, aby mohol svoju voľbu posledný správca prispôsobiť podľa predchádzajúcich hodnôt ostatných správcov a preto hodnoty šifrujeme. Jedná sa o kryptografický protokol nazývaný *commitment scheme* [11][13]. Namiesto asymetrickej kryptografie môžeme použiť aj hashovacie funkcie.

8. Každý správca vypočíta hodnotu $pid = h(n_0|n_1|\ldots|n_{I-1})$ a odošle ju žiadateľovi. Výsledkom je nová identita (pid, $ipAdresa$, k_{pb}) podpísaná všetkými správcami, ktorá bude následne broadcastom zaslaná všetkým uzlom v sieti Liberty Share.

5.5 Alternatívne protokoly

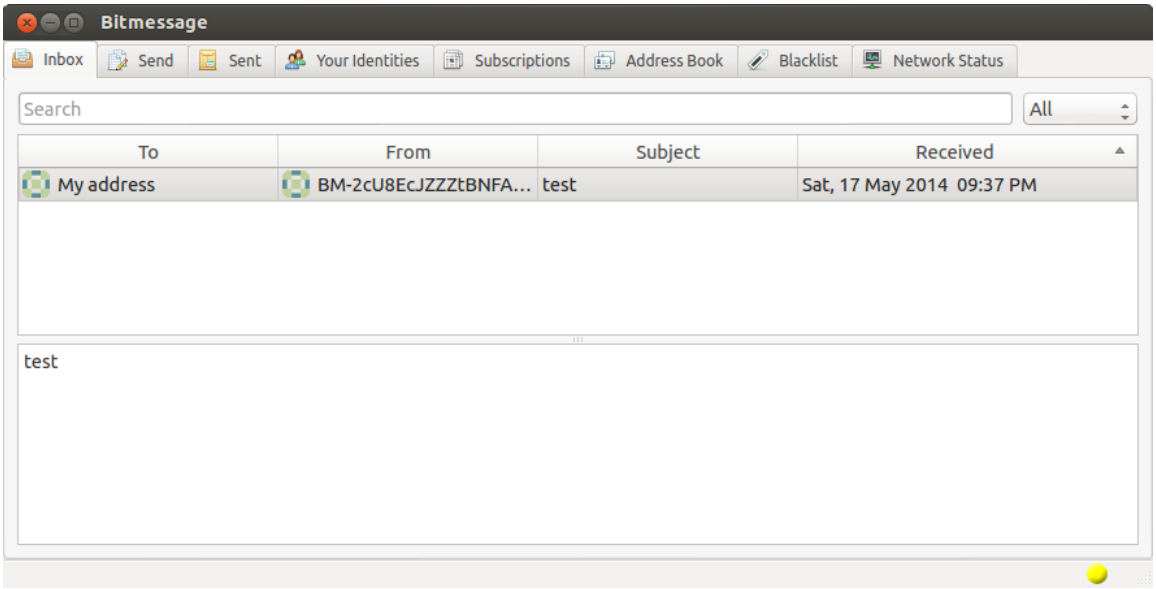
Existuje viacero P2P sietí a aplikácií, ktorých účel je rovnaký alebo podobný systému Liberty Share. V tejto sekcii si tieto aplikácie predstavíme a porovnáme ich bezpečnosť, efektivitu a mieru poskytnutej anonymity vzhľadom k aplikácii Liberty Share.

5.5.1 Bitmessage

Bitmessage je protokol slúžiaci pre bezpečné a anonymné zasielanie správ. Jeho architektúra je založená na protokole Bitcoin, s ktorým zdieľa mnoho spoločných vlastností. Každá správa zaslaná týmto protokolom je šifrovaná privátnym kľúčom autora správy, pričom adresa autora je tvorená hashom verejného kľúča. Táto technika zaručuje bezpečnú identifikáciu autora správy a zabraňuje falšovaniu správ.

Anonymita a necenzúrovateľnosť v aplikácii Bitmessage je zabezpečená pomocou broadcastu v neštrukturovanej P2P sieti. Miera anonymity je v tomto prípade $d_x = 1 - N^{-1}$, ale počet sieťových prenosov potrebných na odoslanie jednej správy je kN, kde k je stupeň uzla. Pretože sieť nepoužíva DHT, tak nie sú správy v sieti ukladané dlhšie ako dva dni. To znamená, že príjemca správy musí byť odpojený zo siete najviac dva dni, inak správa nebude doručená (ale bude neskôr poslaná znovu).

Klient príjemcu správy je povinný potvrdiť prijatie správy. Ak príjemca správu nepotvrdí, tak sa predpokladá, že nie je do siete pripojený a autor správu odošle znovu. Doba medzi predchádzajúcim a nasledujúcim poslaním správy sa exponenciálne zvyšuje po každom poslaní. Prvé poslanie teda nastane po dvoch dňoch, ďalšie po štyroch atď. Pre protokol Bitmessage existuje klient vytvorený vo frameworku Qt. Umožňuje posielanie správ, prijímanie správ, správu identít užívateľa, filtrovanie správ a správu kontaktov (viď obr. 5.3).



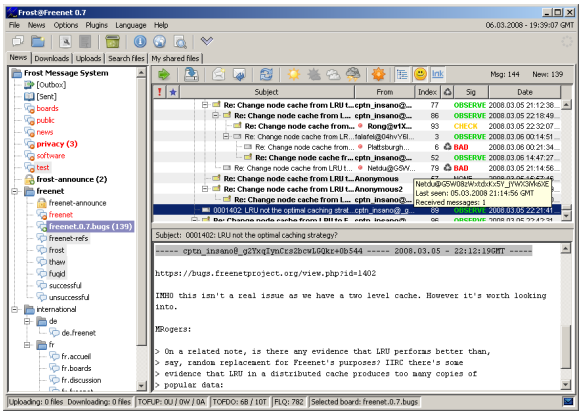
Obr. 5.3: Rozhranie programu Bitmessage

V porovnaní so systémom Liberty Share má Bitmessage oveľa vyššie nároky na prenosové

pásmo ($\log_2(N)$ vs. kN). Teoreticky poskytovaná miera anonymity a necenzúrovateľnosti je rovnaká, avšak Bitmessage má lepšiu odolnosť voči útokom používajúcim spolupracujúce uzly. Slabinou siete Liberty Share je totiž závislosť na Onion Routingu, ktorý zlyháva v prípade, že si autor správy náhodou zvolí cestu cez všetky spolupracujúce uzly útočníka. Zásadným rozdielom oboch sietí je predovšetkým doba archivácie publikovaných správ v sieti, ktorá v prípade siete Libery Share nie je obmedzená.

5.5.2 Freenet

Primárnym cieľom protokolu Freenet nie je anonymné publikovanie textových správy, ale publikovanie ľubovolného obsahu. Avšak, existuje implementácia anonymného posielania správ vybudovaná práve na protokole Freenet.



Obr. 5.4: Implementácia zasielania správ v programe Frost

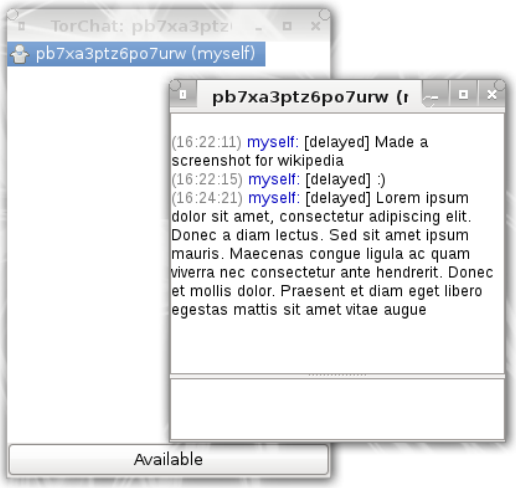
Freenet používa decentralizované úložisko dát v prostredí neštrukturovanej P2P siete. Dáta sú v sieti vyhľadávané spôsobom podobným náhodnej prechádzke, s tým rozdielom, že pri výbere ďalšieho skoku je použitá heuristická funkcia. Konzument obsahu tak nemá možnosť zistiť, či je jeho susedný uzol autorom dát alebo dáta iba preposiela. Implementáciou tohoto protokolu je napríklad software Frost (viď obr. 5.4).

Miera anonymity poskytovaná protokolom Freenet je menšia ako v prípade protokolu Bitmessage, čo je dôsledkom optimalizácie protokolu pre prenos väčších objemov dát ako sú textové správy. V súčasnosti je anonymita Freenetu predmetom verejnej diskusie. Reakciou autorov na niektoré publikované útoky bola zmena zamerania na P2P siet typu darknet[2]. Primárnym rozdielom medzi sieťou Freenet a Liberty Share je usporiadanie siete, kde Freenet používa neštrukturovanú sieť a Liberty Share štrukturovanú. Simulácie smerovania v sieti Freenet vykazuje dobrú efektivitu blízku $O(\log(n)^2)$, pričom však negarantuje nájdenie určitého objektu [18].

5.5.3 TorChat

Aplikácia TorChat slúži pre výmenu krátkych textových správ medzi jej používateľmi, pričom poskytuje anonymitu (ale nie necenzúrovateľnosť). Nejedná sa o protokol ako taký, ale

[2]Neverejná P2P sieť obsahujúca uzly, ktoré si vzájomne dôverujú.



Obr. 5.5: Použitie programu TorChat

o použitie anonymizačnej P2P siete Tor a konceptu *skrytých služieb* (hidden services). Identifikátor užívateľa tvorí adresa skrytej služby. TorChat poskytuje rovnakú mieru anonymity ako anonymizačné protokoly Tor a Onion Routing. Samotná výmena správ však prebieha vždy len medzi dvomi užívateľmi a sieť tak neposkytuje publikačné a archivačné funkcie. Ako môžeme vidieť na obrázku 5.5, grafické používateľské rozhranie je veľmi jednoduché.

Záver

V tejto práci boli podrobne popísane a analyzované základné princípy návrhu P2P sietí. Zvýšená pozornosť bola venovaná predovšetkým technikám zaisťujúcim anonymitu, necenzúrovateľnosť a bezpečnosť. Jednotlivé techniky boli porovnávané z hľadiska ich účinnosti a efektivity, ale taktiež z hľadiska obtiažnosti implementácie v reálnej počítačovej sieti. Zamerali sme sa taktiež na formálne spôsoby hodnotenia poskytovanej miery anonymity.

Teoretické znalosti, ktoré sme nadobudli v prvých troch kapitolách sme ďalej využili pri návrhu P2P protokolu LSPP. Úlohou tohoto protokolu je zabezpečiť efektívnym spôsobom necenzurovateľné a bezpečné publikovanie textových príspevkov v prostredí P2P siete. Návrh tejto siete v našom prípade zabezpečuje taktiež čiastočnú anonymitu používateľov publikačného protokolu.

Protokol LSPP sme implementovali vo frameworku Qt. Výsledkom implementácie je knižnica Liberty Share, ktorá umožňuje vytvárať plnohodnotné aplikácie používajúce navrhnutý protokol. Prednosťou knižnice Liberty Share je predovšetkým možnosť jej použitia na väčšine počítačových a mobilných platformách, ale taktiež efektívna implementácia paralelizmu pomocou asynchrónneho spracovania udalostí. Takýto systém umožňuje obsluhovať súčasne veľké množstvo klientov a jeho efektivita je porovnateľná s výkonnými webovými servermi ako nginx alebo platformou Node.js.

V poslednej kapitole tejto práce sme analyzovali nami navrhnutý protokol LSPP. Bezpečnosť a necenzúrovateľnosť bola v návrhu protokolu zaistená pomocou overených kryptografických algoritmov a preto sme sa zamerali predovšetkým na anonymitu, útoky špecifické pre P2P siete a porovnanie s existujúcimi protokolmi poskytujúcimi podobné vlastnosti. Po vykonaní matematickej analýzy a simulácie sme zistili, že protokol zaisťuje čiastočnú anonymitu, ktorej mieru je možné zvýšiť na úkor poklesu efektivity smerovania. Na základe týchto zistení sme navrhli možné modifikácie protokolu, ktoré by mohli vlastnosti poskytované protokolom LSPP výrazne zlepšiť.

Pokračovanie tejto práce by mohlo viesť viacerými smermi. Jedným z nich by mohol byť prechod na anonymizáciu pomocou broadcastu, ktorá by poskytla lepšiu mieru anonymity a zvýšila by tak výslednú bezpečnosť systému. Kľúčovým problémom by v tejto variante bolo rozdeliť sieť pri dosiahnutí určitej veľkosti na vhodne veľké podsiete, aby sme zabezpečili prijateľnú náročnosť na šírku prenosového pásma. Ďalším možným smerom by bol vývoj plnohodnotnej aplikácie pre počítačové alebo mobilné platformy.

Literatúra

[1] Copyright and Peer-To-Peer Music File Sharing: The Napster Case and the Argument Against Legislative Reform [online]. `http://www.murdoch.edu.au/elaw/issues/v11n1/douglas111.html`, 2004 [cit. 2013-12-25].

[2] Back, A.: A partial hash collision based postage scheme. `http://www.hashcash.org/papers/announce.txt`, 1997 [cit. 2014-05-18].

[3] Baptiste, P.: Attacks on Peer-to-Peer Networks [online]. `http://www.net.t-labs.tu-berlin.de/~stefan/baptiste-pretre.pdf`, 2005 [cit. 2014-05-14].

[4] BIONDI, P.; DESCLAUX, F.: Silver Needle in the Skype [online]. `http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-biondi/bh-eu-06-biondi-up.pdf`, 2006-03-02 [cit. 2013-12-25].

[5] BUFORD, K.; F, J.; YU, H. H.; aj.: *P2P networking and applications*. Morgan Kaufmann, 2009, iSBN 978-0-12-374214-8.

[6] Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, Únor 1981: s. 84–90, iSSN 0001-0782.

[7] Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, ročník 1, 1988: s. 65–75, iSSN 0933-2790.

[8] El-ansary, S.; Alima, L. O.; Brand, P.; aj.: Efficient Broadcast in Structured P2P Networks. `http://www.sics.se/~seif/Publications/paper3.pdf`, 2003 [cit. 2014-05-17].

[9] Karger, D. R.; Ruhl, M.: Simple Efficient Load Balancing Algorithms for Peer-to-peer Systems. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, New York, NY, USA: ACM, 2004, s. 36–43, iSBN 1-58113-840-7.

[10] Kegel, D.: The C10K problem [online]. `http://www.kegel.com/c10k.html`, 2014-02-05 [cit. 2014-04-21].

[11] Lipmaa, H.: Commitment Schemes. `http://www.cs.ut.ee/~lipmaa/crypto/link/commitment/`, 2009 [cit. 2014-05-21].

[12] May, T. C.: The Crypto Anarchist Manifesto [online]. `http://www.activism.net/cypherpunk/crypto-anarchy.html`, 1992 [cit. 2013-12-25].

[13] Menezes, A. J.; Vanstone, S. A.; Oorschot, P. C. V.: *Handbook of Applied Cryptography*. CRC Press, Inc., první vydání, 1996, iSSN 0849385237.

[14] Murdoch, S. J.: The Future of Anonymity and CensorshipResistant Publishing [online]. `http://www.cl.cam.ac.uk/~sjm217/talks/brno08anonymity.pdf`, 2008 [cit. 2014-01-01].

[15] Reed, M.; Syverson, P.; Goldschlag, D.: Anonymous connections and onion routing. *Selected Areas in Communications, IEEE Journal on*, ročník 16, č. 4, 1998: s. 482–494, iSSN 0733-8716.

[16] Reiter, M. K.; Rubin, A. D.: Crowds: Anonymity for Web Transactions. *ACM Trans. Inf. Syst. Secur.*, ročník 1, č. 1, Listopad 1998: s. 66–92, iSSN 1094-9224.

[17] Sandvine: Global Internet Phenomena [online]. `https://www.sandvine.com/trends/global-internet-phenomena/`, 2013 [cit. 2013-12-26].

[18] Shen, X.; Yu, H.; Buford, J.; aj.: *Handbook of Peer-to-Peer Networking*. Springer, 2009, iSBN 978-0387097503.

[19] Steinmetz, R.; Wehrle, K.: *Peer-to-peer systems and applications*. Springer, 2005, iSBN 978-354-0291-923.

[20] Stoica, I.; Morris, R.; Karger, D.; aj.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *SIGCOMM Comput. Commun. Rev.*, ročník 31, č. 4, 2001: s. 149–160, iSSN 0146-4833.

[21] Vasilakos, A.; Parashar, M.; Karnouskos, S.; aj.: *Autonomic Communication*. Springer, 2009, 327-349 s., iSBN 978-0-387-09753-4.

[22] Wright, M. K.; Adler, M.; Levine, B. N.; aj.: The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Trans. Inf. Syst. Secur.*, ročník 7, č. 4, Listopad 2004: s. 489–522, iSSN 1094-9224.

Dodatok A

Dokumentácia správ protokolu LSPP

Protokol LSPP používa na najnižšej vrstve komunikácie zasielanie správ. Táto kapitola obsahuje dokumentáciu štruktúry správ (paketov), zasielaných medzi jednotlivými uzlami a definuje ich sémantiku. Správy sú prenášané protokolom TCP a k ich vytváraniu je využitá serializácia objektov knižnice Liberty Share. Poradie bytov všetkých dátových typov je vždy Big-endian.

Všetky správy sú zložené z nasledujúcich základných typov:

byte – číselná hodnota o veľkosti jeden byte.

byte[] – pole bytov určitej veľkosti.

uint16 – číselná hodnota o veľkosti dva byty bez znamienka.

uint32 – číselná hodnota o veľkosti štyri byty bez znamienka.

Všetky perzistentné dáta v sieti Liberty Share sú ukladané vo forme dátových blokov o maximálnej veľkosti 4 KiB.

Dátový blok má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
dataBlockId	256	Identifikátor dátového bloku.
timestamp	32	Čas vzniku dátového bloku (formát POSIX).
blockLength	32	Dĺžka dátového bloku v bytoch.
blockBytes	0–4096	Obsah dátového bloku.

Tabuľka A.1: Formát dátového bloku

Typická komunikácia prebieha formou požiadavka-odpoveď. Pričom každý paket je odoslaný v nasledujúcom formáte:

Názov	Dĺžka [bit]	Popis
length	32	Udáva dĺžku správy v bytoch.
<message>[1]	0–n	Ľubovoľná správa protokolu LSPP.

Tabuľka A.2: Formát obecného paketu protokolu LSPP

A.1 FIND_SUCCESOR

Touto správou požaduje uzol A od uzla B nájdenie uzla C, ktorý zodpovedá za určitý identifikátor v sieti. Jedná sa o správu zasielanú rekurzívne, až pokiaľ sa príjemcom správy nestane predchodca uzla C. V tom momente posledný príjemca správy vráti odpoveď poslednému odosielateľovi správy atď., až kým sa výsledok nevráti prvému odosielateľovi.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x01`.
objectID	256	Hľadaný identifikátor.

Tabuľka A.3: Požiadavka FIND_SUCCESOR

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
flags	7	Rezervované pre budúce použitie.
nodeEmptyFlag	1	Hodnota 1 znamená, že uzol nebol nájdený a správa ďalej nepokračuje. Hodnota 0 znamená, že uzol bol nájdený a správa pokračuje.
ipAdressVersion	8	Verzia IP adresy. Možné hodnoty: 1. V prípade IPv4 `0x00`. 2. V prípade IPv6 `0x01`. 3. V prípade nepodporovanej verzie `0xFF`.

[1] Notácia <...> znamená, že položka správy je definovaná v samostatnej tabuľke.

ipAdress	32 alebo 128	IP adresa nájdeného uzla. Formát adresy závisí na hodnote ipAdressVersion.
port	16	Port, na ktorom beží RPC server nájdeného uzla.
objectID	256	Identifikátor nájdeného uzla.

Tabuľka A.4: Odpoveď na správu FIND_SUCCESOR

A.2 GET_PREDECESSOR

Touto správou požaduje uzol A od uzla B zaslanie identifikačných údajov o uzle C, ktorý je predchodca uzla B v usporiadaní Chord a teda $C = B$.predecessor. Táto správa sa používa pri stabilizácii siete.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x02`.

Tabuľka A.5: Požiadavka GET_PREDECESSOR

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
flags	7	Rezervované pre budúce použitie.
nodeEmptyFlag	1	Hodnota 1 znamená, že uzol nemá predchodcu a správa ďalej nepokračuje. Hodnota 0 znamená, že uzol má predchodcu a správa pokračuje.
ipAdressVersion	8	Verzia IP adresy. Možné hodnoty: 1. V prípade IPv4 `0x00`. 2. V prípade IPv6 `0x01`. 3. V prípade nepodporovanej verzie `0xFF`.
ipAdress	32 alebo 128	IP adresa predchodcu. Formát adresy závisí na hodnote ipAdressVersion.
port	16	Port, na ktorom beží RPC server predchodcu.
objectID	256	Identifikátor predchodcu.

Tabuľka A.6: Odpoveď na správu GET_PREDECESSOR

A.3 NOTIFY

Správa `NOTIFY` je zasielaná v pravidelných intervaloch a pomáha pri stabilizácii siete. Touto správou dáva uzol A vedieť uzlu B, že môže byť jeho predchodcom v usporiadaní Chord.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x03`.
flags	7	Rezervované pre budúce použitie.
nodeEmptyFlag	1	Táto hodnota by mala byť pri bezchybnej funkcii vždy 1 (zdrojový uzol nemôže byť prázdny).
ipAdressVersion	8	Verzia IP adresy. Možné hodnoty: 1. V prípade IPv4 `0x00`. 2. V prípade IPv6 `0x01`. 3. V prípade nepodporovanej verzie `0xFF`.
ipAdress	32 alebo 128	IP adresa typu loopback podľa špecifikácie IETF. Formát adresy závisí na hodnote ipAdressVersion. Príjemca správy je následne zodpovedný za dosadenie IP adresy odosielateľa (zdrojový uzol nemusí poznať svoju verejnú adresu).
port	16	Port, na ktorom beží RPC server odosielateľa.
objectID	256	Identifikátor odosielateľa.

Tabuľka A.7: Odpoveď na správu NOTIFY

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
-	0	Odpoveď tvorí prázdna správa.

Tabuľka A.8: Odpoveď na správu NOTIFY

A.4 PING

Touto správou zisťuje uzol A prítomnosť uzla B v sieti. Môže byť taktiež použitá k meraniu času odozvy a následnej optimalizácii usporiadania siete a smerovacích tabuliek.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x04`.

Tabuľka A.9: Požiadavka PING

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
-	0	Odpoveď tvorí prázdna správa.

Tabuľka A.10: Odpoveď na správu PING

A.5 GET_ID

Touto správou požaduje uzol A od uzla B zaslanie jeho identifikátora (ObjectID) v rámci siete. Uplatnenie tejto správy nachádzame pri implementácii bezpečnostných funkcií, kedy nemôžeme plne dôverovať informáciám od susedných uzlov, ale požadujeme explicitné zistenie identifikátoru od uzla, ktorému údajne patrí.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x05`.

Tabuľka A.11: Požiadavka GET_ID

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
objectID	256	Identifikátor príjemcu správy.

Tabuľka A.12: Odpoveď na správu GET_ID

A.6 PUT_DATA

Touto správou požaduje uzol A od uzla B nájdenie uzla C, ktorý bude zodpovedať za dátový blok ukladaný do siete. Jedná sa o správu zasielanú rekurzívne, až pokiaľ sa príjemcom správy nestane predchodca uzla C. V tom momente posledný príjemca správy požiada uzol C o uloženie dátového bloku prostredníctvom správy `STORE_DATA`. Uzol A obdrží ako výsledok adresu uzla C.

Rekurzívna implementácia tejto správy je kľúčová k zachovaniu anonymity v sieti. Rekurzívne zasielanie správ so sebou však nesie určitú réžiu a vytvára dlhšie odozvy v sieti.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x06`.
<dataBlock>	320–4416	Dátový blok.

Tabuľka A.13: Požiadavka PUT_DATA

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
flags	7	Rezervované pre budúce použitie.
nodeEmptyFlag	1	Hodnota 1 znamená, že cieľový uzol nebol nájdený a správa ďalej nepokračuje. Hodnota 0 znamená, že cieľový uzol bol nájdený a správa pokračuje.
ipAdressVersion	8	Verzia IP adresy. Možné hodnoty: 1. V prípade IPv4 `0x00`. 2. V prípade IPv6 `0x01`. 3. V prípade nepodporovanej verzie `0xFF`.
ipAdress	32 alebo 128	IP adresa uzla, do ktorého bol dátový blok uložený. Formát adresy závisí na hodnote ipAdressVersion.
port	16	Port, na ktorom beží RPC server uzla zodpovedného za dátový blok.
objectID	256	Identifikátor uzla zodpovedného za dátový blok.

Tabuľka A.14: Odpoveď na správu PUT_DATA

A.7 GET_DATA

Touto správou požaduje uzol A od uzla B nájdenie uzla C, ktorý zodpovedá za požadovaný dátový blok uložený v sieti a zároveň požaduje získanie tohoto dátového bloku. Jedná sa o správu zasielanú rekurzívne, až pokiaľ sa príjemcom správy nestane predchodca uzla C. V tom momente posledný príjemca správy požiada uzol C prostredníctvom správy `LOAD_DATA` o vrátenie dátového bloku. Uzol A obdrží ako výsledok zoznam obsahujúci všetky dátové bloky uložené pod daným ObjectID.

Rekurzívna implementácia tejto správy je kľúčová k zachovaniu anonymity v sieti. Rekurzívne zasielanie správ so sebou však nesie určitú réžiu a vytvára dlhšie odozvy v sieti.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x07`.
dataBlockId	256	Identifikátor dátového bloku.

Tabuľka A.15: Požiadavka GET_DATA

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
listLength	32	Dĺžka zoznamu blokov.
<dataBlock$_1$>	320–4416	Dátový blok.
<dataBlock$_2$>	320–4416	Dátový blok.
...		
<dataBlock$_n$>	320–4416	Dátový blok.

Tabuľka A.16: Odpoveď GET_DATA

A.8 STORE_DATA

Touto správou požaduje uzol A od uzla B uloženie dátového bloku do lokálneho dátového skladu uzla B. Jedná sa o nerekurzívnu správu. Túto správu spravidla zasiela uzol, ktorý reaguje na správu `PUT_DATA` a zároveň pozná adresáta dátového bloku (v našom prípade uzol B). V opačnom prípade by smeroval správu `PUT_DATA` na ďalší uzol a správu `STORE_DATA` by neodoslal.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x08`.
<dataBlock>	320–4416	Dátový blok.

Tabuľka A.17: Požiadavka STORE_DATA

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
-	0	Odpoveď tvorí prázdna správa.

Tabuľka A.18: Odpoveď na správu STORE_DATA

A.9 LOAD_DATA

Touto správou požaduje uzol A od uzla B vrátenie všetkých dátových blokov uložených v lokálnom dátovom sklade uzla B pod zadaným ObjectID. Jedná sa o nerekurzívnu správu. Túto správu spravidla zasiela uzol, ktorý reaguje na správu `GET_DATA` a zároveň pozná adresáta (v našom prípade uzol B), ktorý zodpovedá za dátové bloky uložené pod daným ObjectID. V opačnom prípade by smeroval správu `GET_DATA` na ďalší uzol a správu `LOAD_DATA` by neodoslal.

Požiadavka má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
type	8	Obsahuje konštantu `0x09`.
dataBlockId	256	Identifikátor dátového bloku.

Tabuľka A.19: Požiadavka LOAD_DATA

Odpoveď má nasledujúci formát:

Názov	Dĺžka [bit]	Popis
listLength	32	Dĺžka zoznamu blokov.
$<\text{dataBlock}_1>$	320–4416	Dátový blok.
$<\text{dataBlock}_2>$	320–4416	Dátový blok.
...		

<dataBlock$_n$>	320–4416	Dátový blok.

Tabuľka A.20: Odpoveď LOAD_DATA

Dodatok B

Výsledky simulácie

Táto kapitola obsahuje výsledky simulácie smerovania vo vrstve prepojení Chord. Simulácia používa nedeterministický smerovací algoritmus a metódu Monte Carlo, pričom umožňuje následný výpočet pravdepodobnosti zdrojového uzla x na základe uzla y, ktorý správu doručil uzlu z.

Kód B.1 Simulácia nedeterministického smerovania protokolu Chord v jazyku Python

```
m = 4
N = int(pow(2,m))
messagesCount = 10000

# we want to record a source node (x) and a last node (y)
# that delivered a message to a target node (z)
# we use two dimensional table in order to record a stats data
resultTable = [[0 for x in range(N)] for y in range(N)]
reversedResultTable = [[0 for y in range(N)] for x in range(N)]

# check if a finger is in valid interval
def inIntervalOC(x, a, b):
  if a < b:
    return (a < x and x <= b)
  elif a > b:
    return (a < x or x <= b)
  else:
    return True

# simulate one nondeterministic route of a message
def sendNondet(x, y, z):
  for i in reversed(range(m)):
    if i > 1:
      rand = randrange(- int(pow(2,i-1)), int(pow(2,i-1)))
    else:
      rand = 0
    nextNode = (y + int(pow(2,i)) + rand) % N
    if inIntervalOC(nextNode,y,z):
      if nextNode == z:
        resultTable[x][y] += 1
        reversedResultTable[y][x] += 1
        return
      else:
        sendNondet(x, nextNode, z)
      break

# simulate sending 10 000 messages from every node to node 0
for i in range(1,N):
  for j in range(messagesCount):
    sendNondet(i, i, 0)

# print result table
print("Results:")
for y in reversedResultTable:
  print(y)
```

y	$x = B$	$x = C$	$x = D$	$x = E$	$x = F$	$x = G$	$x = H$	$x = I$	$x = J$	$x = K$	$x = L$	$x = M$	$x = N$	$x = O$	$x = P$
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	146	0	0	0	1250	0	0	0	0	0	0	0	0	0	0
G	180	165	0	0	0	1239	0	0	0	0	0	0	0	0	0
H	146	172	183	0	0	0	1255	0	0	0	0	0	0	0	0
I	154	168	133	133	0	41	0	1230	0	0	0	0	0	0	0
J	188	166	186	151	192	27	77	0	1240	0	0	0	0	0	0
K	226	196	170	182	138	214	70	120	0	1176	0	0	0	0	0
L	639	606	542	473	459	544	611	276	396	0	3185	0	0	0	0
M	877	771	727	634	540	583	749	839	427	554	0	3526	0	0	0
N	527	769	739	670	526	534	554	714	727	411	463	301	2484	0	0
O	2037	1981	2502	2271	2055	1959	1898	1990	2394	2371	1458	2106	1548	6253	0
P	4880	5006	4818	5486	4840	4859	4786	4831	4816	5488	4894	4067	5968	3747	10000

Tabuľka B.1: Výsledky simulácie pre $m = 4$ a maximálnu dĺžku skoku 8

y	$x = B$	$x = C$	$x = D$	$x = E$	$x = F$	$x = G$	$x = H$	$x = I$	$x = J$	$x = K$	$x = L$	$x = M$	$x = N$	$x = O$	$x = P$
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	704	737	697	768	511	985	842	618	626	0	2493	0	0	0	0
M	669	738	756	729	708	508	903	739	620	608	0	2563	0	0	0
N	853	760	862	823	860	789	606	1021	877	684	616	325	2470	0	0
O	2578	2497	2406	2521	2579	2441	2566	2066	3082	2593	1933	2541	1551	6284	0
P	5196	5268	5279	5159	5342	5277	5083	5556	4795	6115	4958	4571	5979	3716	10000

Tabuľka B.2: Výsledky simulácie pre $m = 4$ a maximálnu dĺžku skoku 4

y	$x = B$	$x = C$	$x = D$	$x = E$	$x = F$	$x = G$	$x = H$	$x = I$	$x = J$	$x = K$	$x = L$	$x = M$	$x = N$	$x = O$	$x = P$
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	101	96	100	100	95	97	92	99	95	95	97	100	102	92	101
G	108	108	104	113	112	116	110	99	106	106	103	103	109	106	103
H	115	118	121	115	113	118	114	115	115	115	106	113	117	116	117
I	139	136	134	131	127	133	134	133	130	138	124	131	126	127	129
J	160	157	148	147	145	147	156	152	147	149	154	144	161	154	145
K	178	175	181	173	175	176	178	179	173	180	176	183	180	177	179
L	529	523	516	530	521	525	531	524	526	527	527	526	525	524	524
M	665	658	672	668	647	657	670	677	662	669	663	683	662	652	663
N	616	618	618	611	620	610	612	613	618	608	621	613	625	620	618
O	2170	2177	2190	2201	2208	2210	2217	2204	2195	2172	2224	2189	2193	2188	2200
P	5213	5228	5211	5206	5231	5205	5181	5198	5228	5236	5197	5212	5196	5239	5216

Tabuľka B.3: Výsledky simulácie pre $m = 4$, maximálnu dĺžku skoku 8 a náhodný počiatočný uzol v sieti

Dodatok C

Obsah CD

Priložené CD obsahuje nasledujúce súbory a priečinky:

- `ls-src/` – Zdrojové kódy knižnice Liberty Share.
- `ls-doc/` – Dokumentácie knižnice Liberty Share.
- `simulacia/` – Zdrojový kód simulačného skriptu a výsledky simulácie.
- `sprava-src/` – Zdrojový tvar technickej správy.
- `sprava.pdf` – Technická správa vo formáte PDF.